

(43)公開日 平成15年3月7日(2003.3.7)

(51)Int.Cl. ⁷		識別記号	F I	データベース ⁸ (参考)	
G 0 6 F	3/16	3 2 0	G 0 6 F	3/16	3 2 0 H 5 B 0 1 9
	13/00	5 1 0		13/00	5 1 0 C 5 B 0 8 0
	15/00	3 1 0		15/00	3 1 0 B
	15/02	3 1 0		15/02	3 1 0 D
		3 2 5			3 2 5 B

審査請求 未請求 請求項の数24 OL (全 49 頁) 最終頁に続く

(21)出願番号	特願2002-132053(P2002-132053)	(71)出願人	391056933 マイクロソフト コーポレーション MICROSOFT CORPORATION アメリカ合衆国 ワシントン州 98052-6399 レッドモンド ワン マイクロソフト ウェイ (番地なし)
(22)出願日	平成14年5月7日(2002.5.7)	(74)代理人	10007/481 弁理士 谷 義一 (外2名)
(31)優先権主張番号	60/289,041		
(32)優先日	平成13年5月4日(2001.5.4)		
(33)優先権主張国	米国(US)		
(31)優先権主張番号	09/960,232		
(32)優先日	平成13年9月20日(2001.9.20)		
(33)優先権主張国	米国(US)		
(31)優先権主張番号	10/117,141		
(32)優先日	平成14年4月5日(2002.4.5)		
(33)優先権主張国	米国(US)		

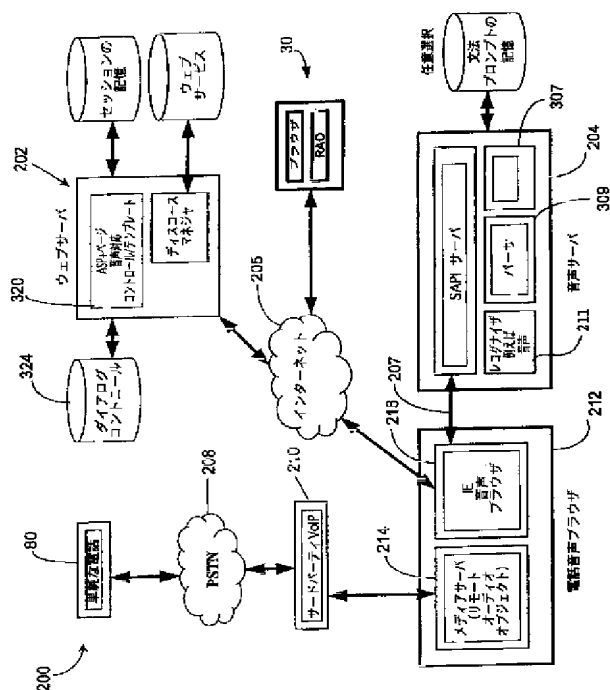
最終頁に続く

(54) 【発明の名称】 ウェブ対応認識アーキテクチャを有するシステムおよびその方法

(57) 【要約】

【課題】 入力認識を提供するのに使用されるウェブ対応認識アーキテクチャを有するシステムに、統一したアーキテクチャを持たせる。

【解決手段】 サーバ／クライアントシステムは、リモートからアクセスすることのできる情報を有するウェブサーバ202を有するネットワークを含む。クライアントデバイスは、マイクロフォンと、スピーカまたはディスプレイなどのレンダリング構成要素とを含む。クライアントデバイスは、ウェブサーバ202から情報入手し、その情報中に含まれる、フィールドと関連付けられた入力データを記録するように構成し、また、認識に使用する文法の指示とともに、入力データを遠隔位置に送信するように適合する。認識サーバ204は、入力データおよび文法の指示を受信し、また、何が認識されたかを示すデータを、クライアントおよびウェブサーバの少なくとも1つに戻す。



【特許請求の範囲】

【請求項1】 データを処理するサーバ／クライアントシステムであって、リモートでアクセスすることのできる情報を有するウェブサーバと、入力すべきフィールドを指示するレンダリングデバイスを有するクライアントデバイスであって、ユーザがその後行う入力の対象とするフィールドを指示すると、前記フィールドの各々と関連付けられた入力データを記録するように構成され、かつ前記入力データを遠隔位置に送信するように適合されたクライアントデバイスと、前記入力データを受信し、何が認識されたかを表すデータを、前記クライアントおよび前記ウェブサーバの少なくとも1つに戻すように構成された認識サーバとを含むネットワークを備えたことを特徴とするシステム。

【請求項2】 前記ウェブサーバから受信し、前記クライアントデバイスに提供する前記情報は、マークアップ言語であることを特徴とする請求項1に記載のシステム。

【請求項3】 前記クライアントデバイスが受信する前記マークアップ言語は、1つまたはいくつかのマークアップ部分、および1つまたはいくつかのスクリプト部分を含むことを特徴とする請求項2に記載のシステム。

【請求項4】 前記マークアップ言語は、文法をフィールドに関連付ける指示を含むことを特徴とする請求項3に記載のシステム。

【請求項5】 前記認識サーバは、前記入力データおよび前記文法の指示を受信することを特徴とする請求項4に記載のシステム。

【請求項6】 前記入力データを前記リモートサーバに送信する前に前記入力データを正規化するように、前記クライアントが適合されることを特徴とする請求項1に記載のシステム。

【請求項7】 前記ウェブサーバは、前記クライアントデバイスのためにマークアップ言語ページを動的に生成するサーバサイドプラグインモジュールを含むことを特徴とする請求項2に記載のシステム。

【請求項8】 マイクロフォンおよびスピーカを有する第2のクライアントデバイスをさらに含み、該第2のクライアントデバイスは、前記ユーザに与えるプロンプトに回答して、各フィールドセットに関連付けられた音声データを記録するように構成され、かつ、前記音声データを前記認識サーバに送信するように適合されることを特徴とする請求項1に記載のシステム。

【請求項9】 前記第2のクライアントデバイスは、前記ウェブサーバからのコンテンツをレンダリングすることのできる電話機および音声ブラウザを含むことを特徴とする請求項8に記載のシステム。

【請求項10】 前記ウェブサーバから受信し、前記クライアントデバイスの各々に提供する前記情報は、マー

クアップ言語であることを特徴とする請求項9に記載のシステム。

【請求項11】 前記ウェブサーバは、前記クライアントデバイスの各々のために動的にマークアップ言語を生成するサーバサイドプラグインモジュールを含むことを特徴とする請求項8に記載のシステム。

【請求項12】 前記サーバサイドプラグインモジュールは、クライアントデバイスのタイプに応じて、マークアップ言語を動的に生成することを特徴とする請求項11に記載のシステム。

【請求項13】 前記サーバサイドプラグインモジュールは、前記クライアントデバイスのタイプを検出することを特徴とする請求項12に記載のシステム。

【請求項14】 前記ウェブサーバは、前記サーバサイドプラグインモジュールからアクセスすることが可能な複数のダイアログモジュールを含み、各ダイアログモジュールは音声認識を使用するデータの入手に関係し、前記サーバサイドプラグインモジュールが、ダイアログモジュールに応じて前記マークアップ言語を生成することを特徴とする請求項11に記載のシステム。

【請求項15】 前記ウェブサーバおよび前記認識サーバを単一のマシンに配置することを特徴とする請求項1に記載のシステム。

【請求項16】 データを処理するサーバ／クライアントシステムであって、

リモートからアクセスすることのできる情報を有するウェブサーバと、

マイクロフォンおよびレンダリング構成要素を有し、対応するフィールドを有する前記情報を前記ウェブサーバから入手するように構成されたクライアントデバイスであって、前記フィールドの各々と関連付けられた入力データを記録するようにさらに構成され、かつ、入力の実行に使用する文法の指示とともに前記入力データを遠隔位置に送信するように適合されたクライアントデバイスと、

前記入力データおよび前記文法の指示を受信するように構成され、何が入力されたかを表すデータを、前記クライアントおよび前記ウェブサーバの少なくとも1つに戻す認識サーバとを含むネットワークを備えたことを特徴とするシステム。

【請求項17】 前記ウェブサーバから受信し、前記クライアントデバイスに提供する前記情報は、マークアップ言語であることを特徴とする請求項16に記載のシステム。

【請求項18】 前記クライアントデバイスは電話機を含み、前記レンダリング構成要素はスピーカを含むことを特徴とする請求項16に記載のシステム。

【請求項19】 前記レンダリング構成要素は、前記フィールドを示すディスプレイを含むことを特徴とする請求項16に記載のシステム。

【請求項20】 前記クライアントおよび前記認識サーバを単一のマシンに配置することを特徴とする請求項16に記載のシステム。

【請求項21】 クライアント／サーバシステムで音声認識を処理する方法であって、
前記クライアントデバイスのユーザから入力データを入手するように構成されたエクステンションを有するマークアップ言語ページを、ウェブサーバからクライアントデバイスに送信するステップと、
前記マークアップ言語ページを前記クライアントデバイスでレンダリングするステップと、
前記ユーザからの入力に応じて、入力データを入手するステップと、
前記入力データおよび関連付けられた文法の指示を、前記クライアントデバイスから遠隔に位置する認識サーバに送信するステップと、
前記クライアントデバイスおよびウェブサーバのうち少なくとも1つで、何が入力されたかを表す認識結果を、前記認識サーバから受信するステップとを備えたことを特徴とする方法。

【請求項22】 前記マークアップ言語をレンダリングするステップは、データ入力用のフィールドを表示するステップを含み、入力データを入手するステップは、その後行う入力をどのフィールドに関連付けるかについての指示を前記ユーザから受信するステップを含むことを特徴とする請求項21に記載の方法。

【請求項23】 前記マークアップ言語をレンダリングするステップは、前記ユーザに対して音声により指示を促すステップを含むことを特徴とする請求項21に記載の方法。

【請求項24】 前記マークアップ言語はスクリプティングを模倣することを特徴とする請求項21に記載の方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、インターネットなどのワイドエリアネットワークを介した情報のアクセスに関する。より詳細には、本発明は、各種の方法を使用してクライアント側で情報およびコントロールを入力することを可能にするウェブ対応認識に関する。

【0002】

【従来の技術】人々が、個人情報マネージャ(PIM)、デバイス、および携帯電話のような小型のコンピューティングデバイスを日常活動で使用する頻度は増す一方である。現在では、こうしたデバイスを作動させるのに使用されるマイクロプロセッサに利用できる処理能力が増大したことにより、これらデバイスの機能性が高まっており、場合によっては機能を一体化している。例えば現在、携帯電話の多くは、アドレス、電話番号などの個人情報の記憶に使用できるだけでなく、インターネットの

アクセスおよびブラウズにも使用することができる。

【0003】こうしたコンピューティングデバイスをインターネットブラウズに使用し、あるいは他のサーバ／クライアントアーキテクチャで使用することから、情報をコンピューティングデバイスに入力することが必要となる。不都合なのは、携行を容易にするためにこうしたデバイスを可能な限り小さくしたいという要求があり、利用可能なコンピューティングデバイス筐体の表面面積が限られているために、アルファベットの全文字を個別のボタンとして備える従来型のキーボードが通例は不可能であることである。

【0004】最近、VoiceXML(音声拡張可能マークアップ言語)の使用によるなどの音声ポータルが進歩し、電話だけを使用してインターネットコンテンツにアクセスすることが可能になっている。このアーキテクチャでは、ドキュメントサーバ(例えばウェブサーバ)が、VoiceXMLインタープリタを通じてクライアントからの要求を処理する。ウェブサーバはそれに応じてVoiceXMLドキュメントを生成することができ、このドキュメントはVoiceXMLインタープリタによって処理し、ユーザに対して音声としてレンダリングされる。ユーザは、音声認識を通じて音声コマンドを使用することにより、ウェブをナビゲートすることができる。

【0005】VoiceXMLは、フロー制御タグを用いるマークアップ言語であるが、フロー制御は、イベントティング(eventing)および個別のスクリプトを含むHTML(ハイパーテキストマークアップ言語)のフロー制御モデルには従わない。VoiceXMLは一般に、電話ベースの音声のみの対話に特に適したフォーム解釈アルゴリズムを含むが、このアルゴリズムでは通例、ユーザから得られる情報をシステムまたはアプリケーションによって制御する。グラフィカルユーザインタフェースも提供し、クライアントーサーバ関係で利用することのできるアプリケーションにVoiceXMLを直接組み込むには、開発者は、2つの形態のウェブオーサリングを習得する必要がある。すなわち、VoiceXMLのオーサリングと、HTML(など)を使用したオーサリングであるが、これらはそれぞれ異なるフロー制御モデルに従っている。

【0006】

【発明が解決しようとする課題】したがって、インターネットなどのサーバ／クライアントアーキテクチャで音声認識を提供するのに使用されるアーキテクチャ、またはその部分、および方法に改良を加えることが現在必要とされている。音声認識用のオーサリングツールは、PIM、電話などの小型のコンピューティングデバイスに容易に適合できなければならない。前述の不利点の1つ、いくつか、またはすべてに対処するウェブオーサリングのアーキテクチャまたは方法が特に必要とされる。

他の入力方法が可能なアーキテクチャも有益であろう。

【0007】本発明は、このような課題に鑑みてなされたもので、その目的とするところは、インターネットなどのサーバ/クライアントアーキテクチャで入力認識を提供するのに使用される、統一したアーキテクチャを備えたウェブ対応認識アーキテクチャを有するシステムおよびその方法を提供することにある。

【0008】

【課題を解決するための手段】データ処理用のサーバ/クライアントシステムは、リモートにアクセスできる情報を含んだウェブサーバを有するネットワークを含む。クライアントデバイスは、マイクロフォンと、スピーカまたはディスプレイなどのレンダリング構成要素を含む。クライアントデバイスは、ウェブサーバから情報を入手して、その情報に含まれるフィールドと関連付けられた入力データを記録するように構成する。クライアントデバイスは、認識に使用する文法の指示とともに入力データを遠隔位置に送信するように適合する。認識サーバは入力データおよび文法の指示を受け取る。認識サーバは、何が入力されたかを示すデータをクライアントおよびウェブサーバの少なくとも1つに戻す。

【0009】

【発明の実施の形態】ウェブベース認識のアーキテクチャおよびその実施方法を説明する前に、このアーキテクチャで機能することが可能なコンピューティングデバイスについて全般的に説明しておくことが有用であろう。本明細書で図1を参照すると、データ管理デバイス(PIM、PDAなど)の例示的形態が30に表されている。ただし、本発明は、下記で論じるこの他のコンピューティングデバイス、特に入力ボタンなどを装備するには表面積が限られたコンピューティングデバイスを使用して実施することも企図している。例えば、電話および/またはデータ管理デバイスも、本発明から利益を受けることができる。このようなデバイスは、既存の携帯個人情報管理デバイスおよびその他の携帯電子デバイスと比較して高いユーティリティを備え、そのデバイスの諸機能とコンパクトなサイズにより、ユーザがデバイスを常に携帯することを促すと思われる。したがって、本明細書に記載するアーキテクチャの範囲は、本明細書に記載する例示的なデータ管理デバイスまたはPIMデバイス、電話機、またはコンピュータの開示によっては制限しないものとする。

【0010】データ管理モバイルデバイス30の例示的な形態を図1に示す。モバイルデバイス30は筐体32を含み、ディスプレイ34を含むユーザインタフェースを有する。ユーザインタフェースには、スタイラス33と合わせて接触感知式の表示画面を使用する。スタイラス33は、指定された座標でディスプレイ34を押す、またはディスプレイ34に接触して、フィールドを選択し、カーソルの開始位置を選択的に移動するのに使用

し、あるいはジェスチャや手書きなどによる他の方法でコマンド情報を提供するのに使用する。これに代えて、あるいはこれに加えて、ナビゲーション用に1つまたは複数のボタン35a、35b、35cをデバイス30上に含むことができる。さらに、回転ホイール、ローラなどの他の入力機構も提供することができる。ただし、本発明は、これらの形態の入力機構によっては制限しないことに留意されたい。例えば、この他の形態の入力には、コンピュータビジョン(vision)を用いるなどの視覚的な入力を含むことができる。

【0011】次いで図2を参照すると、モバイルデバイス30を構成する機能構成要素をブロック図で示している。中央演算処理装置(CPU)50は、ソフトウェア制御機能を実施する。CPU50はディスプレイ34に結合され、制御ソフトウェアに従って生成されるテキストおよびグラフィックアイコンが、ディスプレイ34に表示される。スピーカ43を、通例はデジタルからアナログに変換する変換器59とともにCPU50に結合し、音声による出力を提供することができる。ユーザがモバイルデバイス30にダウンロードまたは入力したデータは、CPU50と双方向に結合した不揮発性の読み出し/書き込みランダムアクセスメモリ記憶装置54に記憶する。ランダムアクセスメモリ(RAM)54は、CPU50が実行する命令の揮発性の記憶、およびレジスタ値など一時的なデータの記憶を提供する。構成オプションや他の変数のデフォルト値は、読み出し専用メモリ(ROM)58に記憶する。ROM58は、モバイル30の基本機能、およびその他のオペレーティングシステムカーネル機能(例えばソフトウェアコンポーネントをRAM54にロードするなど)を制御する、デバイス用のオペレーティングシステムソフトウェアの記憶にも使用することができる。

【0012】RAM54は、アプリケーションプログラムの記憶に使用するPCのハードドライブ機能と同様の方式で、コードの記憶機構としても機能する。不揮発性メモリをコードの記憶に使用しているが、コードは代わりに、コードの実行には使用されない揮発性メモリに記憶することも可能であることに留意されたい。

【0013】無線信号は、CPU50に結合された無線トランシーバ52を通じて、モバイルデバイスによって送信/受信することができる。所望の場合には、コンピュータ(例えばデスクトップコンピュータ)から、あるいは配線式ネットワークから直接データをダウンロードするために、任意選択の通信インタフェース60を提供することもできる。したがって、インタフェース60は、例えば赤外線リンク、モデム、ネットワークカードなど、様々な通信装置の形態を備えることができる。

【0014】モバイルデバイス30は、マイクロフォン29、アナログ/デジタル(A/D)変換器37、および記憶装置54に記憶された任意選択の認識プログラム

(音声、DTMF、手書き、ジェスチャ、またはコンピュータ画像)を含む。一例として、デバイス30のユーザからの音声による情報、命令、またはコマンドにตอบสนองして、マイクロフォン29が音声信号を提供し、それをA/D変換器37でデジタル化する。音声認識プログラムは、デジタル化した音声信号に正規化および/または特徴抽出機能を行って、中間の音声認識結果を得る。無線トランシーバ52または通信インタフェース60を使用して、下記で説明し、図5のアーキテクチャに表すリモートの認識サーバ204に音声データを送信する。その後認識結果をモバイルデバイス30に戻して、そこでレンダリング(例えば視覚的かつ/または可聴的に)を行い、最終的にウェブサーバ202(図5)に送信するが、本明細書でウェブサーバ202とモバイルデバイス30はクライアント/サーバ関係で動作している。これと同様の処理を、他の形態の入力にも使用することができる。例えば、手書き入力を、デバイス30での前処理により、または前処理によらずにデジタル化することができる。音声データと同様に、この形態の入力も認識のために認識サーバ204に送信することができ、認識結果が、デバイス30および/またはウェブサーバ202の少なくともどちらかに戻される。同様に、DTMFデータ、ジェスチャデータ、および視覚データも同じように処理することができる。入力形態に応じて、デバイス30(および下記で説明する他の形態のクライアント)は、カメラや視覚入力など必要なハードウェアを含む。

【0015】図3は、携帯電話80の一例示の実施形態の平面図である。電話機80は、ディスプレイ82およびキーパッド84を含む。一般に、図2のブロック図は図3の電話機にも該当するが、他機能を行うために必須の追加回路が必要になることもある。例えば、図3の実施形態には、電話としての動作に必須のトランシーバが必要になるが、このような回路は本発明には関連しない。

【0016】上記の携帯式またはモバイル型のコンピュータティングデバイス以外にも、本発明は、一般的なデスクトップコンピュータなど数多くの他のコンピュータティングデバイスにも使用できることが理解されよう。例えば、身体能力が限られたユーザにとって完全な英数文字キーボードなど他の従来型の入力装置の操作が困難である場合に、本発明は、そのようなユーザがコンピュータまたは他のコンピュータティングデバイスにテキストを入力することを可能にする。

【0017】本発明はまた、数多くの他の汎用または特殊目的のコンピュータティングシステム、環境、または構成での動作が可能である。本発明とともに使用するのに適した周知のコンピュータティングシステム、環境、および/または構成の例には、従来型の(regular)電話(画面を備えない)、パーソナルコンピュータ、サーバコンピュータ、携帯用デバイスまたはラップトップ

デバイス、マルチプロセッサシステム、マイクロプロセッサベースのシステム、セットトップボックス、プログラマブル家庭用電化製品、ネットワークPC、ミニコンピュータ、メインフレームコンピュータ、上記のシステムまたはデバイスなどのうち任意のものを含む分散型コンピュータティング環境が含まれるがこれらに限定するものではない。

【0018】以下は、図4に示す汎用コンピュータ120の簡単な説明である。ただし、この場合もコンピュータ120は、適切なコンピュータティング環境の一例に過ぎず、本発明の使用または機能性の範囲に関して何らの制限を示唆するものではない。また、コンピュータ120は、この図に示す構成要素のいずれか、またはその組合せに関する依存性や要件を有するものとも解釈すべきではない。

【0019】本発明は、プログラムモジュールなどコンピュータで実行するコンピュータ実行可能命令の一般的な状況で説明することができる。一般に、プログラムモジュールには、特定タスクを実行する、または特定の抽象データタイプを実施する、ルーチン、プログラム、オブジェクト、コンポーネント、データ構造などが含まれる。本発明はまた、通信ネットワークを通じてリンクした遠隔処理装置によってタスクを実行する分散型コンピュータティング環境でも実施することができる。分散型コンピュータティング環境では、プログラムモジュールは、メモリ記憶装置を含む、ローカルおよびリモートどちらのコンピュータ記憶媒体に置いてよい。以下で、図面の助けを借りて、プログラムおよびモジュールによって実行するタスクを説明する。当業者は、この説明および図面をプロセッサ実行可能命令として実施することができ、この命令はどの形態のコンピュータ読み取り可能な記録媒体にも書き込むことができる。

【0020】図4を参照すると、コンピュータ120の構成要素には、プロセッサ140、システムメモリ150、およびシステムメモリを含む各種システム構成要素をプロセッサ140に結合するシステムバス141が含まれるが、これらに限定しない。システムバス141は、メモリバスまたはメモリコントローラ、周辺バス、および各種バスアーキテクチャのうち任意のものを使用したローカルバスを含む数種のバス構造のうち任意のものでよい。このようなアーキテクチャには、例えば、ISA(Industry Standard Architecture)バス、USB(Universal Serial Bus)、MCA(Micro Channel Architecture)バス、EISA(Enhanced ISA)バス、VESA(Video Electronics Standards Association)ローカルバス、およびメザニンバスとしても知られるPCI(Peripheral Component Interconnect)バ

スが含まれるがこれらに限定するものではない。コンピュータ120は、通例、各種のコンピュータ読み取り可能な記録媒体を含んでいる。コンピュータ読み取り可能な記録媒体は、コンピュータ120からアクセスすることができる任意の利用可能な媒体でよく、これには揮発性および不揮発性媒体、リムーバルおよび取外し不能媒体が含まれる。例えば、コンピュータ読み取り可能な記録媒体は、コンピュータ記憶媒体および通信媒体を含むことができるがこれらに限定しない。コンピュータ記憶媒体には、コンピュータ可読命令、データ構造、プログラムモジュール、またはその他のデータなどの情報を記憶するための任意の方法または技術に実施された、揮発性および不揮発性、リムーバルおよび取外し不能媒体が含まれる。コンピュータ記憶媒体には、RAM、ROM、EEPROM、フラッシュメモリ、または他のメモリ技術、CD-ROM、デジタル多用途ディスク（DVD）、またはその他の光ディスク記憶、磁気カセット、磁気テープ、磁気ディスク記憶または他の磁気記憶装置、あるいは所望の情報の記憶に使用することができ、コンピュータ120からアクセスすることが可能な任意の他の媒体が含まれるがこれらに限定するものではない。

【0021】通信媒体は、通例、搬送波または他の搬送機構などの変調データ信号中のコンピュータ可読命令、データ構造、プログラムモジュール、または他のデータを実施し、また任意の情報伝達媒体を含む。用語「変調データ信号」とは、情報を信号中に符号化するような方式で、その特性の1つまたは複数を設定または変更した信号を意味する。例として、通信媒体には、配線式ネットワークまたは直接配線接続などの配線式媒体、および音響、FR、赤外線、および他の無線媒体などの無線媒体が含まれるが、これらに限定しない。上記の媒体の任意の組合せも、コンピュータ読み取り可能な記録媒体の範囲に含むものとする。

【0022】システムメモリ150は、読み出し専用メモリ（ROM）151およびランダムアクセスメモリ（RAM）152などの揮発性および／または不揮発性メモリの形態でコンピュータ読み取り可能な記録媒体を含む。起動時などにコンピュータ120中の要素間の情報の転送を助ける基本ルーチンを含んだ基本入出力システム153（BIOS）は、通例ROM151に記憶する。RAM152は、通例、プロセッサ140から即座にアクセスすることができ、かつ／またはプロセッサ140が現在操作しているデータおよび／またはプログラムモジュールを含む。例として、図4にはオペレーティングシステム154、アプリケーションプログラム155、他のプログラムモジュール156、およびプログラムデータ157を示しているが、これらに限定しない。

【0023】コンピュータ120は、他のリムーバル／取外し不能、揮発性／不揮発性のコンピュータ読み取り

可能な記録媒体も含むことができる。図4には、取外し不能、不揮発性の磁気媒体との読み出しまたは書き込みを行うハードディスクドライブ161、リムーバル、不揮発性の磁気ディスク172との読み出しまたは書き込みを行う磁気ディスクドライブ171、およびCD-ROMや他の光媒体などのリムーバル、不揮発性の光ディスク176との読み出しまたは書き込みを行う光ディスクドライブ175を示すが、これらは例にすぎない。この例示的動作環境で使用できる、この他のリムーバル／取外し不可能、揮発性／不揮発性のコンピュータ記憶媒体には、磁気テープカセット、フラッシュメモ리카ード、デジタル多用途ディスク、デジタルビデオテープ、ソリッドステートRAM、ソリッドステートROMなどが含まれるがこれらに限定しない。ハードディスクドライブ161は、通例、インタフェース160など取外し不能のメモリインタフェースを通じてシステムバス141に接続し、磁気ディスクドライブ171および光ディスクドライブ175は通例、インタフェース170などのリムーバルメモリインタフェースによってシステムバス141に接続する。

【0024】上記で説明し、図4に示すドライブおよびそれに関連するコンピュータ記憶媒体は、コンピュータ120のコンピュータ可読命令、データ構造、プログラムモジュール、およびその他のデータの記憶を提供する。例えば、図4では、ハードディスクドライブ161は、オペレーティングシステム164、アプリケーションプログラム165、他のプログラムモジュール166、およびプログラムデータ167を記憶するものとして示している。これらのコンポーネントは、オペレーティングシステム154、アプリケーションプログラム155、他のプログラムモジュール156、およびプログラムデータ157と同じものでも、異なるものでもよいことに留意されたい。本明細書では、オペレーティングシステム164、アプリケーションプログラム165、他のプログラムモジュール166、およびプログラムデータ167が少なくとも異なるコピーであることを示すために、これらに異なる番号を与えている。

【0025】ユーザは、キーボード182、マイクロフォン183、およびマウスやトラックボール、タッチパッドなどのポインティングデバイス181などの入力装置を通じて、コンピュータ120にコマンドおよび情報を入力することができる。この他の入力装置（図示せず）には、ジョイスティック、ゲームパッド、衛星放送受信アンテナ、スキャナなどが含まれる。これらの入力装置およびこの他の入力装置は、多くの場合、システムバスに結合したユーザ入力インタフェース180を通じてプロセッサ140に接続するが、パラレルポート、ゲームポート、あるいはユニバーサルシリアルバス（USB）など他のインタフェースおよびバス構造によって接続することもできる。モニタ184または他種の表示装

置も、ビデオインタフェース185などのインタフェースを介して、システムバス141に接続する。コンピュータは、モニタ以外にも、スピーカ187およびプリンタ186など他の周辺出力装置も含むことができ、これらは出力周辺インタフェース188を通じて接続することができる。

【0026】コンピュータ120は、リモートコンピュータ194など1つまたは複数のリモートコンピュータへの論理接続を使用するネットワーク化環境で動作することができる。リモートコンピュータ194は、パーソナルコンピュータ、携帯用デバイス、サーバ、ルータ、ネットワークPC、ピアデバイス、または他の一般的なネットワークノードでよく、通例は上記でコンピュータ120との関連で説明した要素の多くまたはすべてを含む。図4に示す論理接続には、ローカルエリアネットワーク(LAN)191およびワイドエリアネットワーク(WAN)193が含まれるが、この他のネットワークを含んでもよい。このようなネットワーキング環境は、オフィス、企業規模のコンピュータネットワーク、イントラネット、およびインターネットで一般的に見られる。

【0027】LANネットワーキング環境で使用する場合、コンピュータ120は、ネットワークインタフェースすなわちアダプタ190を通じてLAN191に接続する。WANネットワーキング環境で使用する場合、コンピュータ120は通例モデム192か、またはインターネットなどのWAN193を介して通信を確立するための他の手段を含む。モデム192は、内蔵型でも外付け式でもよく、ユーザ入力インタフェース180または他の適切な機構を介してシステムバス141に接続することができる。ネットワーク環境では、コンピュータ120との関連で図示するプログラムモジュール、またはその一部をリモートのメモリ記憶装置に記憶することができる。例として図4に、リモートアプリケーションプログラム195をリモートコンピュータ194に常駐するものとして示しているが、これに限定しない。図のネットワーク接続は例示的なものであり、コンピュータ間に通信リンクを確立する他の手段を使用してよいことは理解されよう。

【0028】図5に、本発明で実施することのできるウェブベース認識のアーキテクチャ200を示す。一般に、ウェブサーバ202に記憶された情報には、モバイルデバイス30(本明細書では、入力の状態に基づき、適宜、表示画面、マイクロフォン、カメラ、タッチセンシティブパネルなどを有する他形態のコンピューティングデバイスをも表す)を通じて、または情報を音声により、またはキーを押すのに応答して電話機80が生成するトーンを通じて要求する電話機80を通じてアクセスすることができる。電話機の場合には、ウェブサーバ202からの情報を音声のみによりユーザに提供する。

【0029】より重要なのは、情報をデバイス30を通じて得るか、または音声認識を用いて電話機80を通じて得るかに関係なく、単一の認識サーバ204がどちらの動作モードもサポートすることができる点でアーキテクチャ200が統一されていることである。さらに、アーキテクチャ200は、周知のマークアップ言語(例えばHTML、XHTML、cHTML、XML、WMLなど)の拡張を使用して動作する。したがって、ウェブサーバ202に記憶された情報には、これらのマークアップ言語で使用される周知のGUI方式を用いてアクセスすることもできる。周知のマークアップ言語の拡張を使用することにより、ウェブサーバ202でのオーサリングが容易になり、現在存在するレガシーアプリケーションも、音声認識を含むように容易に修正することができる。

【0030】一般に、デバイス30は、ウェブサーバ202が提供するHTMLページ、スクリプトなどを実行する。一例として、音声(voice)認識が必要な場合には、デジタル化したオーディオ信号または音声特徴などの音声データ(オーディオ信号は上記のようにデバイス30で前処理する)を、音声認識中に使用する文法または言語モデルの指示とともに、認識サーバ204に提供する。認識サーバ204の実施態様は多くの形態をとることが可能であり、そのうちの1つを図示したが、一般にはレコグナイザ211を含む。認識の結果は、所望の場合、または適切な場合にはローカルのレンダリングのためにデバイス30に戻される。認識と、使用する場合には任意のグラフィカルユーザインタフェースとを通じて情報を編集すると、必要な場合には、デバイス30はその情報をウェブサーバ202に送信し、そこでさらに処理を行い、さらにHTMLページ/スクリプトを受信する。

【0031】図5に示すように、デバイス30、ウェブサーバ202、および認識サーバ204は共通に(commonly)接続されており、また本明細書ではインターネットなどのワイドエリアネットワークであるネットワーク205を通じて個別にアドレス指定することができる。したがって、これらの装置はいずれも物理的に相互に近接して配置する必要はない。特に、ウェブサーバ202が認識サーバ204を含む必要はない。この方式によると、ウェブサーバ202におけるオーサリングを、それが行うべきアプリケーションに集中させることができ、オーサ(author)は認識サーバ204の複雑性を知る必要がない。認識サーバ204は、独自に設計してネットワーク205に接続することができ、それによりウェブサーバ202でさらに変更を行わなくとも更新および改良することができる。下記で説明するように、ウェブサーバ202は、クライアント側のマークアップおよびスクリプトを動的に生成することのできるオーサリング機構も含むことができる。別の実施形態で

は、実装マシンの能力に応じて、ウェブサーバ202、認識サーバ204、およびクライアント30を組み合わせることができる。

【0032】例えば、クライアントがパーソナルコンピュータなどの汎用コンピュータを含む場合には、クライアントは認識サーバ204を含むことができる。同様に、所望の場合には、ウェブサーバ202および認識サーバ204を単一マシンに組み込むことが可能である。

【0033】クライアントデバイスに関して、クライアント／サーバシステムで入力データを処理する方法は、クライアントデバイスのユーザから入力データを得るように構成された拡張を有するマークアップ言語ページをサーバから受信することと、クライアントデバイスでマークアップ言語ページを実行することと、入力データ（ユーザから得た音声、DTMF、手書き、ジェスチャ、または画像を表す）およびそれに関連する文法をクライアントからリモートに位置する認識サーバに送信することと、認識サーバからの認識結果をクライアントで受信することを含む。クライアント／サーバシステムのクライアントデバイスで実行するマークアップ言語を有するコンピュータ読み取り可能な記録媒体を提供することができ、このマークアップ言語は、そのクライアントデバイスで入力される入力データと関連付ける文法を指示する命令を有する。

【0034】電話機80を通じたウェブサーバ202へのアクセスには、配線式または無線式の電話網208への電話機80の接続が含まれ、この電話網が電話機80をサードパーティのゲートウェイ210に接続する。ゲートウェイ210は、電話機80を電話音声ブラウザ212に接続する。電話音声ブラウザ212は、電話インタフェースを提供するメディアサーバ214と、音声ブラウザ216を含む。デバイス30と同様に、電話音声ブラウザ212は、ウェブサーバ202からHTMLページ／スクリプトなどを受信する。より重要なのは、これらのHTMLページ／スクリプトが、デバイス30に提供されるHTMLページ／スクリプトと同様の形態であることである。この方式によると、ウェブサーバ202は、デバイス30と電話機80を個別にサポートする必要がなく、さらには標準的なGUIクライアントを個別にサポートする必要もない。むしろ、共通のマークアップ言語を使用することができる。さらに、デバイス30と同様に、電話機80から送信される可聴信号からの音声認識は、ネットワーク205、または例えばTCP/IPを使用する専用回線207を通じて、音声ブラウザ216から認識サーバ204に提供される。ウェブサーバ202、認識サーバ204、および電話音声ブラウザ212は、図4に示す汎用デスクトップコンピュータなど任意の適切なコンピューティング環境に実施することができる。

【0035】ただしDTMF認識を用いる場合は、この

形態の認識は、一般的には認識サーバ204ではなくメディアサーバ214で行うことに留意されたい。すなわち、メディアサーバでDTMF文法を使用することになる。

【0036】HTML、XHTML、cHTML、XML、WMLなどのマークアップ言語、または他のSGML由来のマークアップを用いるマークアップ言語は、クライアント／サーバアーキテクチャで認識を提供するコントロールおよび／またはオブジェクトを含むことができる。この方式では、オーサが、このようなアーキテクチャで使用される主流のウェブ開発プラットフォームであるこれらのマークアップ言語に、すべてのツールと専門知識を活用することができる。

【0037】一般に、コントロールおよび／またはオブジェクトには、次の機能の1つまたは複数を含むことができる。レコグナイザの構成、レコグナイザの実行、および／または後処理のためのレコグナイザコントロールおよび／またはオブジェクト；シンセサイザの構成およびプロンプト再生のためのシンセサイザコントロールおよび／またはオブジェクト；入力文法リソースを指定する文法コントロールおよび／またはオブジェクト；および／または、認識結果を処理するためのバインドコントロールおよび／またはオブジェクト。拡張は、軽量のマークアップレイヤになるように設計し、これにより、聴覚、視覚、手書きなどによるインタフェースの能力を既存のマークアップ言語に付加する。したがって、拡張は次のものには依存しない。例えばHTMLなど拡張が含まれる高レベルページ；例えばテキストから音声へのフォーマットや文法フォーマットなど、言語リソースへのリファレンスにその拡張が使用した低レベルフォーマット；および認識サーバ204で使用する認識プラットフォーム、および音声合成プラットフォームの個々の属性。

【0038】認識に適したコントロールおよび／またはオブジェクトを有するマークアップを説明する前に、本発明でHTMLマークアップ言語とともに実施する簡単なGUIの例を考察しておくと思われる。図6を参照すると、簡単なGUIインタフェースは、オンライン販売を完了するためにクレジットカード情報をウェブサーバに提出することを含む。この例では、クレジットカード情報は、Visa、MasterCard、あるいはAmerican Expressなど、使用するクレジットカードの種類を入力するフィールド250を含む。第2のフィールド252はクレジットカード番号の入力を可能にし、第3のフィールド254は有効期限の入力を可能にする。フィールド250、252、および254に入力した情報を送信するための「提出」ボタン264が提供される。

【0039】図7は、クライアントから上述のクレジットカード情報を得るためのHTMLコードを示してい

る。一般に、このような形態のマークアップ言語で一般的なように、コードは本体部分260とスクリプト部分262を含む。本体部分260は、実行するアクションのタイプ、使用するフォーム、各種の情報フィールド250、252、および254を指定するコードの行を含み、また提出ボタン264（図6）用のコードも含む。この例は、イベントサポートと、埋め込まれたスクリプトホスティングも表しており、提出ボタン264が起動されると、スクリプト部分262で関数「verify」が呼び出され、または実行される。「verify」関数は、各クレジットカード（Visa、MasterCard、American Express）のカード番号の長さが適切な長さであることを確認する。

【0040】図8は、音声認識を使用してウェブサーバ204に提供するクレジットカード情報を得るための、図6と同じGUIを生成するクライアントマークアップを表している。音声認識については下記で図8～16との関連で説明するが、本明細書で説明する技術は、手書き認識、ジェスチャ認識、および画像認識にも同様に応用できることを理解されたい。

【0041】一般に、エクステンション（拡張；一般には「タグ」としても知られる）はXML要素の小セットであり、関連する属性およびDOMオブジェクトプロパティ、イベント、およびメソッドを含み、ソースマークアップドキュメントと合わせて使用することにより、認識インタフェース、DTMFまたは呼制御をソースページに適用する。エクステンションの形式（formality）および意味（semantics）はソースドキュメントの性質に依存しないので、エクステンションは、HTML、XHTML、cHTML、XML、WMLで、あるいは任意の他のSGML由来のマークアップとともに等しく効果的に使用することができる。エクステンションは、階層的にすることが可能な新しい機能オブジェクトまたは要素を提供するドキュメントオブジェクトモデルに従う。各要素については付録中で詳細に説明するが、一般に、要素には属性、プロパティ、メソッド、イベント、および／または他の「子」要素を含むことができる。

【0042】本明細書で、エクステンションは、ブラウザを実行するデバイスの機能に応じて、異なる2つの「モード」で解釈できることにも留意されたい。第1のモード「オブジェクトモード」では、全機能を利用することができる。アプリケーションによるエクステンションのプログラム上の操作は、そのデバイスのブラウザが使用可能にするどの機構でも実行することができる。これには、例えば、XHTMLブラウザにおけるJavaScriptインタプリタや、WMLブラウザにおけるWMLScriptインタプリタなどがある。この理由から、エクステンションのコアプロパティおよびメソッドの小セットだけを定義すればよく、これらは、デバイス

すなわちクライアント側に存在する任意のプログラム機構によって操作される。オブジェクトモードは、イベントリングおよびスクリプティングを提供し、またより多くの機能を提供して、ダイアログのオーサに、音声対話に対するより細かなクライアント側におけるコントロールを与えることができる。本明細書で使用する場合、フルイベントおよびスクリプティングをサポートするブラウザを「アップレベルブラウザ」と呼ぶ。この形のブラウザは、エクステンションのすべての属性、プロパティ、メソッド、およびイベントをサポートする。アップレベルブラウザは、通例、より高い処理能力を持つデバイスで使用される。

【0043】エクステンションは、「宣言モード」でもサポートすることができる。本明細書で使用する場合、宣言モードで動作するブラウザを「ダウンレベルブラウザ」と呼び、これは完全なイベントリングおよびスクリプティング機能はサポートしない。代わりにこの形のブラウザは、所与のエクステンションの宣言的側面（すなわちコア要素および属性）をサポートするが、DOM（ドキュメントオブジェクトモデル）オブジェクトのプロパティ、メソッド、およびイベントのすべてはサポートしない。このモードは専ら宣言構文だけを用い、さらに、SMIL（同期化マルチメディア統合言語）2.0などの宣言マルチメディア同期化および協調機構（同期マークアップ言語）と併せて使用することができる。ダウンレベルブラウザは、通例、処理能力が限られたデバイスで使用される。

【0044】ここで、特定の入力モードについて論じておきたい。詳細には、音声認識を、少なくともディスプレイと併せて、そして別の実施形態ではポインティングデバイスとも併せて使用して、データ入力フィールドを指定すると特に有用である。具体的には、このモードのデータ入力では、ユーザは一般に、いつフィールドを選択し、それに対応する情報を提供するかを制御することができる。例えば、図6の例では、ユーザはまずフィールド252にクレジットカード番号を入力し、次いでフィールド250にクレジットカードの種類を入力し、最後にフィールド254に有効期限日を入力することができる。同様に、ユーザは、所望の場合にはフィールド252に戻り、誤った入力を訂正することもできる。下記で説明するように音声認識と組み合わせると、平易で自然なナビゲーション形態が提供される。本発明で使用する場合、フィールドを自由な形で選択することを可能にする画面表示と、音声認識の両方を使用するこの形の入力を「マルチモーダル」と呼ぶ。

【0045】再び図8を参照すると、HTMLマークアップ言語のコードが示されている。図7に示すHTMLコードと同様に、このコードも、本体部分270およびスクリプト部分272を含んでいる。また図7に示すコードと同様に、図8に示すコードは、フォームの位置お

および実行するアクションのタイプに関する指示を含む。フィールド250、252、および254それぞれへの情報の入力、各々コード部分280、282、および284によって制御または実行する。最初にコード部分280を参照すると、例えばデバイス30のスタイラス33を使用してフィールド250を選択すると、イベント「onClick」が開始され、これによりスクリプト部分272の関数「talk」が呼び出されるか、または実行される。このアクションは、一般にフィールド250に予想されるデータタイプと関連付けられた、音声認識で使用する文法を起動する。複数の入力技術（例えば音声とペンクリック／ローラ）を使用するこの種の対話を「マルチモーダル」と呼ぶ。

【0046】図8に例示する音声認識エクステンションは、クライアントのブラウザにおいてデフォルトの視覚表現を有さないことに留意されたい。これは、多くのアプリケーションでは、オーサが、アプリケーション仕様のグラフィック機構をソースページで使用するにより、ページの各種コンポーネントの音声使用可能を知らせることを想定しているためである。それでも、視覚的な表現が望ましい場合には、エクステンションをそのように修正することができる。

【0047】再び文法を参照すると、この文法は、文脈自由文法、N文法、ハイブリッド文法などの構文文法であるがこれらに限定しない。（言うまでもなく、それに対応する形態の認識を利用する際には、DTMF文法、手書き文法、ジェスチャ文法、および画像文法を使用する。本明細書で使用する場合、「文法」とは認識を行うための情報を含み、別の実施形態では、例えば特定のフィールドに入力されることが予想される入力に対応する情報を含む。）マークアップ言語の最初のエクステンションを含む新しいコントロール290（本明細書では「reco」と識別する）は様々な要素を含むが、そのうち2つを図に示す。すなわち文法要素「grammar」と「bind」要素である。一般に、ウェブサーバ202からクライアントにダウンロードするコードと同様に、文法はウェブサーバ202を発信元とし、クライアントにダウンロードするか、かつ／または音声処理のためにリモートサーバに転送することができる。文法は次いで、そのキャッシュでローカルで記憶することができる。最終的に、文法は認識に使用するために認識サーバ204に提供する。文法要素は、インライン文法、または属性を使用して参照する文法を指定するのに使用する。

【0048】認識を行った音声、手書き、ジェスチャ、画像などに対応する認識結果を認識サーバ204から受け取ると、recoコントロール290の構文を提供してそれに対応する結果を受け取り、それを対応フィールドと関連付けるが、これにはその中のテキストをディスプレイ34でレンダリングすることを含んでもよい。本

明細書に例示する実施形態では、音声認識が終了し、結果をクライアントに送り返すと、クライアントはrecoオブジェクトを非活動化して、認識済みのテキストをそれに対応するフィールドと関連付ける。コード部分282および284もこれと同様に動作し、フィールド252および254ごとに固有のrecoオブジェクトおよび文法を呼び出し、認識されたテキストを受け取ると、それをフィールド252および254とそれぞれ関連付ける。カード番号フィールド252の受信については、関数「handle」が、上記で図7との関連で説明したのと同様の方式で、カードの種類からカード番号の長さを確認する。

【0049】一般に、アーキテクチャ200およびクライアント側のマークアップ言語と併せた音声認識の使用は、次のように行われる。まず、与える音声と関連付けられたフィールドを指示する。図の実施形態ではスタイラス33を使用するが、本発明はスタイラス33の使用に限定するものではなく、ボタン、マウスポインタ、回転ホイールなど任意形態の指示を使用できることは理解されよう。周知のように、視覚的なマークアップ言語を使用して、「onClick」などそれに対応するイベントを提供することができる。本発明は、音声、手書き、ジェスチャなどのコマンドの開始を指示するのに、「onClick」イベントの使用だけに限定しない。「onSelect」など、任意の利用可能なGUIも同じ目的に使用することができる。一実施形態では、このようなイベントは、それに対応する音声の開始および／または終わりの両方を示す役割を果たすので、特に有用である。また、音声の対象とするフィールドは、ユーザの対話を追跡するブラウザ上で実行されるプログラムによっても、ユーザによっても指定することに留意されたい。

【0050】ここで注意したいのは、異なる音声認識シナリオには、認識サーバ204の異なる振る舞いおよび／または出力が必要となることである。認識プロセスの開始はすべての場合に標準的なものであり、すなわちアップレベルブラウザからの明示的なstart（ ）の呼び出しであり、あるいはダウンレベルブラウザでは宣言的な<reco>要素であるが、音声認識を中止する手段は異なる可能性がある。

【0051】上記の例では、マルチモーダルアプリケーションのユーザは、例えば圧力を感知するディスプレイを軽く叩き、接触状態を保持することにより、デバイスへの入力を制御する。するとブラウザは、例えば「pen-up」などのGUIイベントを使用して、認識をいつ中止するかを制御し、その後それに対応する結果を戻す。ただし、電話アプリケーション（下記で説明する）あるいは手を使用せずに済むアプリケーションといった音声のみのシナリオでは、ユーザはブラウザに対する直接的な決定権は一切持たず、認識サーバ204またはク

クライアント30が、いつ認識を中止して結果を戻すか（通例は、文法中のパスを認識した時点）を決定する責任を負わなければならない。さらに、認識を中止する前に中間の結果を戻す必要があるディクテーションや他のシナリオ（「オープンマイクロフォン」としても知られる）の場合には、明示的な中止機能が必要とされるだけでなく、認識プロセスを中止する前に複数の認識結果をクライアント30および／またはウェブサーバ202に戻す必要もある。

【0052】一実施形態では、Reco要素は、下記の3つの認識モードを区別する「mode」属性を含むことができ、これにより認識サーバ204に、いつどのように結果を戻すかを命令する。結果を戻すことは、「onReco」イベントを提供する、または「bind」要素を適宜起動することを意味する。一実施形態では、モードを指定しない場合、デフォルトの認識モードは「自動」にすることができる。

【0053】図14は、音声認識の「自動」モードの動作を図式的に表したものである（他の形態の認識にもこれと同様のモード、イベントなどを提供することができる）。スケジュール281は、認識サーバ204にいつ認識の開始283を指示するか、認識サーバ204がどこで音声を検出し（285）、その音声を終了したこと（287）を判定するかを表している。

【0054】Reco要素の各種の属性は、認識サーバ204の振る舞いを制御する。属性「initialTimeout」289は、認識の開始283から音声の検出285までの間の時間である。この期間を超えると、「onSilence」イベント291が認識サーバ204から提供され、認識が中止されたことを知らせる。認識サーバ204が、発声が認識不可能であると識別した場合は、「onNoReco」イベント293を発行するが、これも認識を中止したことを示す。

【0055】認識を中止またはキャンセルすることができる他の属性には、「babbleTimeout」属性295があるが、これは285の音声の検出後に認識サーバ204が結果を戻さなければならない期間である。この期間を超えると、エラー発生の有無に応じて異なるイベントが発行される。例えば、例外的に発声が長い場合など、認識サーバ204がなおオーディオの処理を行っている場合は、「onNoReco」属性293を発行する。しかし他の何らかの理由で「babbleTimeout」属性295を超えた場合は、認識エラーの可能性が高くなり、「onTimeout」イベント297が発行される。同様に「maxTimeout」属性299も提供することができ、これは、認識の開始283から結果をクライアント30に戻すまでの期間である。この期間を超えると、「onTimeout」イベント297が発行される。

【0056】ただし、「endSilence」属性3

01以上の期間を超えた場合、これは認識が完了していることを示唆するが、この場合は認識サーバ204が自動的に認識を中止し、その結果を戻す。認識サーバ204は、信頼度の測定を実施して、認識結果を戻すべきかどうかを判定することに留意されたい。信頼度の測定値が閾値を下回る場合は、「onNoReco」属性293を発行し、一方信頼度の測定値が閾値を上回る場合は、「onNoReco」属性303および認識結果を発行する。したがって図14は、「自動モード」で、明示的なstop（ ）の呼び出しが行われていない状況を表している。

【0057】図15は、認識サーバ204の「シングルモード」の動作を図式的に表したものである。「自動モード」との関連で上記で説明した属性およびイベントを適用することができ、したがって同じ参照番号で示している。しかし、この動作モードでは、stop（ ）呼び出し305を、スケジュール281上に示している。stop（ ）呼び出し305は、ユーザによる「ペンアップ」などのイベントに相当する。この動作モードでは、認識結果を戻すことは、明示的なstop（ ）呼び出し305によって制御される。すべての動作モードの場合と同じく、「onSilence」イベント291は、「initialTimeout」期間289内に音声を検出されない場合に発行されるが、この動作モードでは認識を中止しない。同様に、stop（ ）呼び出し305以前の認識不可能な発声によって生成される「onNoReco」イベント293によっても認識は中止されない。ただし、「babbleTimeout」属性295または「maxTimeout」属性299と関連付けられた期間を超えた場合は、認識を中止する。

【0058】図16は、認識サーバ204の「複数モード」の動作を図式的に表している。上記で指摘したように、この動作モードは、「オープンマイクロフォン」またはディクテーションのシナリオで使用する。一般に、この動作モードでは、明示的なstop（ ）呼び出し305が受け取られるか、または「babbleTimeout」属性295または「maxTimeout」属性299に関連付けられた期間を超えて、間隔を置いて認識結果を戻す。ただし、「onSilence」イベント291、「onReco」イベント303、または「onNoReco」イベント293のいずれかが発生すると、これらによって認識は中止されないが、「babbleTimeout」期間および「maxTimeout」期間のタイマがリセットされることに留意されたい。

【0059】一般に、この動作モードでは、stop（ ）呼び出し305が受け取られるまで、認識されるフレーズごとに、「onReco」イベント303を発行し、結果を戻す。認識不可能な発声のために「onS

「silence」イベント291が発行された場合は、これらのイベントを報告するが、認識は継続する。

【0060】上記で触れたように、フィールドに関連付けられた1つまたは複数の`reco`オブジェクトを起動するが、これには、少なくともどの文法を使用するかについての指示を認識サーバ204に提供することが含まれる。この情報は、クライアント30で記録して認識サーバ204に送信した音声データを伴うことができる。上記で指摘したように、音声データは、ユーザが入力した音声に関連づけられたストリーミングデータを含むことができ、あるいは音声認識中に使用する音声の特徴を示す、前処理済みの音声データを含むことができる。別の実施形態では、クライアント側の処理に音声データの正規化も含むことができ、認識サーバ204が受け取る音声データが、クライアントごとに比較的均質になるようにする。これにより認識サーバ204の音声処理が簡略化され、認識サーバを、クライアントおよび通信経路のタイプにステートレスにすることができるので、認識サーバ204のスケラビリティをより容易にすることができる。

【0061】認識サーバ204から認識結果を受け取ると、その認識結果を対応するフィールドと関連付け、必要な場合はクライアント側で確認またはチェックを行うことができる。現在クライアントがレンダリングしているコードと関連付けられたすべてのフィールドを完了すると、アプリケーション処理のためにその情報をウェブサーバ202に送信する。前述の内容から、ウェブサーバ202は、認識に適したコードまたはページ/スクリプトをクライアント30に提供しているが、認識サービスはウェブサーバ202によって行われず、認識サーバ204によって行われることが明白であろう。ただし、本発明は、認識サーバ204をウェブサーバ202とまとめて配置する、または認識サーバ204をクライアント30の一部とするような実施を排除するわけではない。すなわち、本明細書で提供するエクステンションは、認識サーバ204をウェブサーバ202またはクライアント30と組み合わせた場合でも有用である。これは、エクステンションが、これら構成要素間に単純かつ利便なインタフェースを提供するからである。

【0062】図8に示す実施形態には示していないが、`reco`コントロールは、適切な音声データを認識サーバ204に導くためのリモートオーディオオブジェクト(RAO)も含むことができる。RAOをプラグインオブジェクトにすることによる利益は、サウンドインタフェースが異なる可能性が高いことから、異なるデバイスまたはクライアントそれぞれに異なるRAOを可能にすることである。さらに、リモートオーディオオブジェクトにより、複数の`reco`要素を同時に起動することが可能になる。

【0063】図9および10は、本発明でページ/スク

リプトを含むHTMLとして実施する音声のみによるマークアップ言語を示す。図に明瞭に示すように、このコードも本体部分300およびスクリプト部分302を含んでいる。マークアップ言語の別のエクステンション、すなわちバージンなどの属性を含むプロンプトコントロール303がある。ただし、図9および10の音声のみの実施形態では、音声認識を別の方式で行う。この場合は、プロセス全体を、未入力(`unfilled`)のフィールドを判定し、かつそれに対応するプロンプトおよび新しいオブジェクトを起動するスクリプト関数「`checkFilled`」によって制御する。しかし、上記で図8との関連で説明したのと同じコンテキストを使用して文法を起動し、音声データおよび使用する文法の指示を認識サーバ204に提供する。同様に、認識サーバ204から受け取った出力を、クライアント(この場合は電話音声ブラウザ212)のフィールドと関連付ける。

【0064】一般に音声のみのアプリケーションに固有の他の機能は、音声認識されなかった際にユーザにそれを知らせることである。図8のようなマルチモーダルアプリケーションでは、「`onNoReco`」は、表示されるフィールドに単にヌル値を入れて、認識が行われなかったことを示すので、それ以上の動作は必要とされない。音声のみの実施形態では、「`onNoReco`」305は関数「`mumble`」を呼び出し、または実行する。この関数は、単語のフレーズを認識サーバ204に転送し、このフレーズは適切なテキストから音声に変換するシステム307(図5)を使用して音声に変換される。認識サーバ204は、オーディオストリームを電話音声ブラウザ212に戻し、次いでユーザが聴くためにそれを電話機80に送信する。同様に、音声のみのアプリケーションに実施するこの他の波形プロンプトも、必要な場合には認識サーバ204によりオーディオストリームに変換する。

【0065】この例では、関数「`welcome`」を介して`welcome`プロンプトを再生すると、関数「`checkFilled`」がユーザに各フィールドを指示し、適切な文法を起動する。これには、入力されたフィールドを反復して、その情報が正しいことを確認することが含まれ、また「`confirmation`」文法の起動が含まれる。この実施形態では、各`reco`コントロールは、先の例の本体部分ではなくて、スクリプト部分302から開始されることに留意されたい。

【0066】マークアップ言語は、異なるタイプのクライアントデバイス(例えば、マルチモーダル、および電話機のような非表示式、音声入力ベースのクライアントデバイス)で実行することができ、各クライアントデバイスと対話するウェブサーバのために、音声に関連するイベント、GUIイベント、および電話イベントのうち少なくとも1つを統一する。これは、ウェブサーバアプ

リケーションのかなりの部分を、汎用的に、あるいはクライアントデバイスのタイプに依存せずに書くことを可能にするので特に有用である。「handle」関数を含む一例を図8、および図9、10に示す。

【0067】図9、10には示していないが、このマークアップ言語には、電話機能をサポートするエクステンションがさらに2つある。すなわち、DTMF（デュアルトーン変調周波）制御と、呼制御の要素またはオブジェクトである。DTMFは、recoコントロールと同様の働きをする。これは、キーパッドストリングからテキスト入力への単純な文法マッピングを指定する。例えば、「1」は食料品部門を意味し、「2」は薬品部門を意味するなどである。一方、呼オブジェクトは、呼の転送や第三者の呼出しのような電話機能を扱う。属性、プロパティ、メソッド、イベントについては付録で詳細に説明する。

【0068】図11および12は、音声のみの動作モードに適したマークアップ言語のさらに別の例を示す。この実施形態では、ユーザは、情報をいつ入力するか、または話すかに関してある程度の制御権を有することができる。言い換えると、このシステムでは、発話を開始させるか、あるいはその他の方法で発話を開始するようにユーザに指示することができるが、ユーザは当初要求されるよりも多くの情報を提供することができる。これは、「混合主導型」の一例である。一般に、この形のダイアログ対話では、ユーザはダイアログの主導権をシステムと分かち合うことができる。上記で触れ、下記で詳細に説明する、ユーザがプロンプトに要求されるよりも多くの情報を提供する場合のほかにも、ユーザはその指示がないときにタスクを切り替えることもできる。

【0069】図11および12の例では、「do_field」と識別する文法は、文法「g_card_types」、「g_card_num」、および「g_expiry_date」と関連付けられた情報を含む。この例では、電話音声ブラウザ212は、「onReco」として示す認識済みの音声を受け取ると、電話機80から受け取った音声データと、「do_field」文法の使用の指示を認識サーバ204に送信し、関数「handle」が呼び出され、または実行されるが、これには音声データから認識されたフィールドの一部またはすべての値を関連付けることが含まれる。すなわち、認識サーバ204から得る結果は、各フィールドについての指示も含んでいる。この情報は構文解析し、405で指定されるバインド規則に従って対応するフィールドと関連付ける。図5に示すように、認識サーバ204はパーサ309を含むことができる。

【0070】図7、8、9、10、11、および12から、非常に類似したウェブ開発フレームワークを使用する。データの提示も、これらの各場合で非常に類似している。さらに、データ提示とフロー制御を分離すること

により、異なるアプリケーション（システム主導型と混合主導型）間、または異なるモダリティ間（GUIウェブベース、音声のみ、およびマルチモーダル）での再使用性を最大限にすることができる。また、これにより、電話機がディスプレイおよびデバイス30と同様の機能を含む場合に、音声のみの動作から電話、そしてマルチモーダル動作への自然な拡張が可能になる。付録Aでは、以上で説明したコントロールおよびオブジェクトの詳細をさらに提供する。

【0071】上記で指摘したように、アップレベルブラウザは、上記の例で認識結果を割り当てるために関数「handle」を起動するなど、各種のニーズを実行するためにスクリプティングを使用することができる。上記で説明し、付録Aの2.1.2にさらに説明する実施形態では、「bind」要素は認識結果を構文解析し、値を割り当てるが、この「bind」要素は「reco」要素の下位要素または子要素である。

【0072】スクリプティングは有用でありうるが、多くの者は、例えばセキュリティ問題などから必ずしも最良のブラウザ実装形態であるとは限らないと見ている。したがって、本発明のさらに別の実施形態または態様では、「bind」要素は（「reco」同様の）高レベル要素であり、他のより豊富なプロパティとともに提供され、実際、それ自体ではスクリプティングを用いずにスクリプティングを実際に模倣することができる。

【0073】スクリプティングを用いない場合、あるいは下記で述べる本発明の態様を使用しない場合、高度なダイアログ効果など下記で述べる機能の一部は、ページを再度ウェブサーバ202に提出し、そこでアプリケーションロジックを実行して新しいページを生成し、そのページを再びクライアントデバイスに送信することによってのみ実現することができる。本発明のこの態様により、プログラマは、サーバへのラウンドトリップを招く（incur）ことなく、そのページのオブジェクトのメソッドを起動することができる。

【0074】上記の実施形態では、「bind」要素は、認識結果をフォーム中またはウェブページ中のフィールドに割り当てるための属性「TargetElement」および「TargetAttribute」しか有さない。別の実施形態では、「bind」要素は、オブジェクトメソッドの起動のために加える「TargetMethod」も含む。「TargetMethod」の使用および機能は、スクリプティングの模倣にとって非常に重要な技術である。例えば、次の構文を使用して、オブジェクト「OBJ1」の「X」メソッドを起動することができる。<bind TargetElement = "OBJ1" TargetMethod = "X" ...>ここに示す例はHTML/XHTMLのイベント構文に従っているが、当業者にとっては、<bind>の使用を一般化して、他のイベント機構を使用することは平易であることに留意された

い。他のイベント機構には、W3Cドキュメントオブジェクトモデルレベル2またはレベル3のイベント機構、ECMA共通言語基盤(CLI)イベントモデル、Java(登録商標)プログラミング言語イベントモデル、W3C同期マルチメディア統合言語(SMIL)、および近く登場するW3CのXMLイベント規格提案が含まれるが、これらに限定するものではない。

【0075】図17および18は、クライアント、特にダウンレベルブラウザで実行可能なマークアップ言語のページである。この例では、音声プロンプトを通じてユーザに希望する飲料を尋ねている。このシステムは次いで、どの飲料が注文されたかを確認する。認識結果に応じて、「bind」要素は、宣言した論理を使用して実行を導く。飲料を確認すると、そのフォームをウェブサーバ202に再度提出するが、これらにスクリプティングは一切用いない。

【0076】一般に、図17および18のマークアップ例は、データ部分350、音声部分352、およびユーザインタフェース部分354、356、および358を含む。部分354は、全般的な質疑から、ユーザが希望する飲料についての認識結果を受け取り、対話式認識フローを誘導して、クリームや砂糖が必要かどうかについて再度指示を促し、尋ねるか、または注文された飲料を確認する。詳細には、部分356は、クリームや砂糖も注文された場合にはその認識結果を受け取る。部分358は、飲料の確認についての認識結果を受け取る。部分360は、新しいメッセージングオブジェクト「SMEX」を用いる呼び出し部分である。「SMEX」については下記でさらに説明する。

【0077】上記で指摘したように、本発明のこの態様の「bind」要素はオブジェクトメソッドの起動を含み、これは、「welcome」オブジェクトの「start」メソッドを361で実行する際に「welcome」プロンプトを再生することにより、図17および18の例でユーザ対話を開始する。

【0078】次いで、362で「asked」オブジェクトの「start」メソッドを実行することにより、ユーザに「ご希望はコーラ、コーヒー、それともオレンジジュースですか?」と尋ねる。次いで、363で、認識「reco_drink」オブジェクトの「start」メソッドを起動することにより認識を実行する。

【0079】次いで部分354のマークアップを実行するが、ここで認識サーバ204が使用する文法は、XPathステートメント「./drink types」によって提供される。この例ではW3CのXPath言語を利用しているが、この概念を、他の標準的言語に拡張することは当業者にとって平易であることに留意されたい。他の標準的言語には、W3CによるXMLクエリ言語(XQL)を含むが、これに限定するものではない。「bind」要素364によって明確に示すよう

に、認識サーバ204から受け取った認識結果の信頼度スコアが10未満である場合は、366でプロンプトオブジェクト「reprompt」を実行し、それに続いてプロンプトオブジェクト「ask」を368で実行し、この時に認識オブジェクト「reco_drink」を370で再度開始する。戻された認識結果が「coffee」で、それが10を超える信頼度を有する場合、372でフィールド「drink」に認識結果の値を割り当て、374でプロンプトオブジェクト「cream_sugar」により、クリームあるいは砂糖を希望するかしないかについてユーザに指示を促す。次いで、376で、部分356の認識オブジェクト「reco_cream_sugar」を起動する。そうでなく、認識結果が信頼度スコアは10を超えるがコーヒーでない場合は、378でフィールド「drink」に再度値を割り当てる。認識結果の確認は、プロンプトオブジェクト「confirm」を実行し、それに続いて部分358の認識オブジェクト「reco_yesno」を382で起動することにより、380で提供する。ユーザが「yes」と答え、その信頼度スコアが10を超える場合は、384でプロンプトオブジェクト「thanks」を再生し、次いで386でフォームを提出する。そうでなく、ユーザが「no」と答えた場合、あるいは認識結果の信頼度スコアが10未満の場合は、390でプロンプトオブジェクト「retry」を実行し、その後再度プロンプトオブジェクト「ask」を392で実行し、「reco_drink」認識オブジェクトを394で起動する。

【0080】上の例から、「bind」要素により、部分354、356、または358で示すような複数のメソッド起動が可能になる。所望の場合は、認識済み結果の複数の割り当ても宣言することができる。ここで説明する実施形態では、複数の割り当ておよびメソッド起動を宣言する場合、それらはドキュメントの順序で実行する。

【0081】別の実施形態では、メソッドの引き数を渡すための規則も提供される。すなわち、一部のメソッドは引き数のリストを必要とする場合がある。これは「arg」下位要素を使用して実現する。例えば、次のマークアップの場合、

```
<bind TargetElement = "OBJ" TargetMethod = "F"><arg>X</arg><arg>Y</arg></bind>
```

は、「OBJ.F(X,Y)」に等しい。すなわち「OBJ」は、パラメータすなわち引き数「X」および「Y」を用いるメソッド「F」を有するオブジェクトである。

【0082】「bind」要素は「event」属性も含むことができ、これはそのバインド要素が対象とするイベントを宣言する。例えば、マークアップ

```
<bind event = "onNoReco" = TargetElement = "prompt
```

```
1" TargetMethod = "start"/>
```

は、「onNoReco」イベントを送る際に、オブジェクト「prompt1」のメソッド「start」を起動することを意味する。例えば図8との関連で上記で説明したように、「bind」要素を「Reco」要素の子要素として使用するのに整合するように、「bind」要素のデフォルト属性は「onReco」にする。

【0083】高レベル要素である「bind」要素は、付録の節2.4に明記するイベントをいずれも含むことができる。さらに、「bind」要素は、アクセスしてプログラムフローを指示するのに使用できる「status」属性を有する「onError」イベントも含むことができる。「bind」要素の他のイベントが「status」属性を有する限り、これらにもアクセスすることができる。

【0084】認識結果の状態の確認に加えて、実行中の現在のドキュメントまたはページも確認することができる。詳細には、「test」および「value」の両属性を拡張して、それを含むドキュメントのルートノードを参照する「host」プリミティブを含ませることができる。例えば、再び図17および18を参照すると、ここに含まれる例は、ユーザがコーヒーを注文した際にクリームあるいは砂糖を希望するかどうかを尋ねる追加の論理を部分354に有する。クリームや砂糖を加え、したがって部分356を起動するためのフラグは、マークアップ「host()/get_drink/drink='coffee'」の指定によって飲料フィールドが「コーヒー」である場合にのみオンになる。

【0085】また、「bind」要素は音声サーバ204からの認識結果、値の受取り、およびそのドキュメント中への割り当てに適用できるだけでなく、メッセージオブジェクト（ここでは「smex」と表す。例えばクライアントデバイスで実行するアプリケーションからの）にも適用できることに留意されたい。図17および18の例では、クライアントデバイスで実行される電話アプリケーションが呼を検出すると、このページが実行される。部分360で、「bind」要素は、メッセージ「/Call_connected」を受け取ると、「welcome」プロンプトを実行または再生し、「reco_drink」オブジェクトを実行することにより認識を開始する。音声サーバ204から受け取る認識結果と同様に、受け取るメッセージも大きく異なる可能性がある。メッセージの一部は、所望のプログラムフローを開始するために明確に規定する。受け取って処理することのできるメッセージもある（例えば、認識サーバから受け取る認識結果と同様に構文解析を行う）。例えば、これにより、キーボードから入力するテキストの自然言語パーサのようにマークアップを使用できるようになる。付録Aのreco要素は、この機能を実行するためのプロパティを含んでいる。同様に、プロンプト

要素を使用し、付録Aでさらに説明するプロパティ「innerText」を使用することにより、動的コンテンツまたはオーディオウェーブファイル用のテキストメッセージを提供することができる。イベントリングは、認識結果のためのイベントリングと同様のものでよい。例えば、イベントリングは「onReceived」を含むことができるが、これは、メッセージソース（例えばクライアントデバイスで実行するアプリケーション）が、ブラウザで使用できるメッセージを有する際に送られる。

【0086】このように、「smex」すなわちメッセージオブジェクトにより、ここに述べるようなマークアップタグを、クライアントデバイスで実行される他のコンポーネントまたはアプリケーションに拡張することが可能になる。別の例として、このメッセージオブジェクトを使用して、クライアントデバイスで実行される聴覚障害者用のTTYコンポーネントと通信することができる。TTYコンポーネントは、音声認識を使用するのではなく、ユーザが入力した内容のメッセージを提供する。このメッセージはその後、認識結果を認識サーバから受け取った場合と同様に使用する。すなわち、メッセージを構文解析して、フォームのフィールドに割り当てるか、あるいは上記の「reco」、「grammar」、または「bind」要素を使用して他の処理を行うことができる。このメッセージまたは「smex」オブジェクトについては、付録Aでさらに説明する。

【0087】「bind」要素は「for」属性も含むことができ、これにより、その動作をページ上の他のオブジェクトに付することができる。例えば次のマークアップ

```
<bind for = "prompt1" event = "onComplete" targetElement = "prompt2" =targetMethod = "start"/>
```

は、オブジェクト「prompt1」がイベント「onComplete」を送ると、オブジェクト「prompt2」のstartメソッドを起動する。

【0088】再び図5を参照すると、ウェブサーバ202は、サーバ側のプラグイン宣言オーサリングツールすなわちモジュール320を含むことができる（例えば、マイクロソフト社によるASPまたはASP+、あるいはJSPなど）。サーバ側のプラグインモジュール320は、クライアント側のマークアップと、さらにはウェブサーバ202にアクセスするクライアントのタイプについて固有形態のマークアップも動的に生成することができる。クライアント情報は、クライアント/サーバ関係が最初に確立されたときにウェブサーバ202に提供することができる。ウェブサーバ202は、クライアントの機能を検出するモジュールまたはルーチンを含むことができる。この方式で、サーバ側のプラグインモジュール320は、それぞれの音声認識シナリオ、すなわち電話機80を通じた音声のみ、あるいはマルチモーダル型

のデバイス30に対する、クライアント側のマークアップを生成することができる。一貫性のあるクライアント側モデルを使用することにより（各アプリケーションで使用できる `reco` およびプロンプトコントロール）、多数の異なるクライアントのアプリケーションオーサリングが大幅に容易になる。

【0089】クライアント側マークアップの動的な生成に加えて、図8、9および10のマークアップ例を用いた、図6に示すようなクレジットカード番号の入手などの高レベルのダイアログモジュールは、アプリケーションオーサリングで開発者が使用するために、記憶装置324に記憶するサーバ側コントロールとして実施することができる。一般に、高レベルダイアログモジュール324は、開発者が指定するパラメータに基づいて、音声のみおよびマルチモーダルの両シナリオで、クライアント側のマークアップおよびスクリプトを動的に生成する。高レベルダイアログモジュールは、開発者のニーズに適合するクライアント側のマークアップを生成するためのパラメータを含むことができる。例えば、クレジットカード情報のモジュールは、クライアント側のマークアップスクリプトが許可すべきクレジットカードの種類を指定するパラメータを含むことができる。サーバ側プラグインモジュール320で使用するASP+ページの例を図13に示す。

【0090】本発明について好ましい実施形態を参照して説明したが、当業者は、本発明の趣旨および範囲から逸脱せずに、形態および詳細を変更することが可能であることを理解されよう。

【0091】付録A

1 概要

以下のタグは、ドキュメントが音声を入力媒体または出力媒体として使用することを可能にするマークアップ要素のセットである。これらのタグは、HTML、XHTML、cHTML、SMIL、WMLなど任意のSGML由来のマークアップ言語に埋め込むことのできる独立型（self-contained）XMLになるように設計されている。本発明で使用するタグは、ワシントン州レッドモンドのマイクロソフト社から入手可能な周知の方法であるSAPI5.0に類似する。タグ、要素、イベント、属性、プロパティ、戻り値などは例示的なものに過ぎず、制限的なものと考えるべきではない。本明細書では音声およびDTMFの認識の場合の例を示すが、同様のタグは他の形の認識にも提供することができる。

【0092】本明細書で論じる主な要素は以下である。
`<prompt...>` 音声合成の構成およびプロンプトの再生

`<reco...>` レコグナイザの構成、認識の実行、および後処理

`<grammar...>` 入力文法リソースの指定

`<bind...>` 認識結果の処理

`<dtmf...>` DTMFの構成および制御

【0093】2 Reco

Reco要素は、可能なユーザ入力と、入力結果の処理手段とを指定するのに使用する。したがって、その主要な要素は`<grammar>`および`<bind>`にすることができ、またレコグナイザプロパティを構成するためのリソースを含む。

【0094】Reco要素は、アップレベルブラウザではStartおよびStopのメソッドを介してプログラマ的に、またはSMILを使用できるブラウザではSMILコマンドを使用して起動する。この要素は、ダウンレベルブラウザ（すなわちスクリプトをサポートしないブラウザ）では、それがページ上にあることにより宣言的にアクティブであると見なす。複数の文法を並行して起動することができるように、複数のReco要素を同時にアクティブと見なすことができる。

【0095】Recoは特定のモード、すなわち「自動」「シングル」または「複数」をとることもでき、これによりそれが使用可能にする認識シナリオの種類と、認識プラットフォームの振る舞いを区別する。

【0096】2.1 Recoの内容

Reco要素は、1つまたは複数の文法と、任意選択で、認識結果を調べ、関連性のある部分をそれを含むページ中の値にコピーするバインド要素のセットとを含む。

【0097】アップレベルブラウザでは、Recoは、プログラマ的な起動、および個々の文法規則の非活動化をサポートする。指定しない場合は、ある認識コンテキストについて、文法のすべての最上位の規則がアクティブになることにも留意されたい。

【0098】2.1.1 <grammar>要素

文法要素は、インラインの、またはsrc属性を使用して参照する文法を指定するのに使用する。通例は少なくとも1つの文法（インラインまたは参照）を指定する。インライン文法はテキストベースの文法形式にすることができるのに対し、参照文法は、テキストベースまたはバイナリタイプにすることができる。複数の文法要素を指定することが可能である。複数の文法要素を指定する場合は、文法の規則を追加規則として同じ文法中に追加する。同じ名前の規則がある場合にはそれに上書きする。

【0099】属性：

・src：インライン文法を指定する場合は任意選択。含める文法のURI。指定しない場合は、ある認識コンテキストについて、文法のすべての最上位規則がアクティブになることに留意されたい。

【0100】・langID：任意選択。音声エンジンが使用する言語を指示する文字列。文字列の形式は、xml:lang定義に従う。例えば、lang

ID=“en-us”は、米国英語を表す。この属性は、langIDを文法URI中で指定しないときにのみ有効である。指定しない場合は、米国英語を使用する。

【0101】langIDが複数の箇所で指定される場合、langIDは、最低の有効範囲からの優先順位に従う。すなわち、リモートの文法ファイル（つまりその文法ファイル中で指定される言語ID）、次いで文法要素、次いでreco要素の順となる。

```
<grammar src="FromCity.xml" />
```

または

```
<grammar>
  <rule toplevel="active">
    <p>から</p>
    <ruleref name="cities" />
  </rule>
  <rule name="cities">
    <l>
      <p>ケンブリッジ</p>
      <p>シアトル</p>
      <p>ロンドン</p>
    </l>
  </rule>
</grammar>
```

srcで参照する文法とインライン文法の両方を指定す

```
<sml confidence="40">
  <travel text="シアトルからボストンまで行きたい">
    <origin_city confidence="45"> シアトル
  </origin_city>
    <dest_city confidence="35"> ボストン
  </dest_city>
  </travel>
</sml>
```

文法中(in-grammar)認識は、セマンティックマークアップ言語すなわちSMLでXMLドキュメントを生成することになっているので、SMLドキュメントからバインドする値は、XPathクエリを使用して参照する。また、値をバインドするページ中の要素（これはフォームコントロールである可能性が高い）は一意に識別すべきなので、これらのターゲット要素は直接参照する。

【0104】属性：

・targetElement：必須。SMLからvalueの内容を割り当てる要素（W3C SMIL 2.0と同様）。

【0105】・targetAttribute：任意選択。SMLからvalueの内容を割り当てるターゲット要素の属性（SMIL 2.0のattributeName属性と同様）。指定しない場合は、「value」になる。

る場合は、インライン規則を参照規則に加え、同じ名前の規則があればそれに上書きする。

【0102】2.1.2 <bind>要素

バインド要素は、認識結果の値をページ中にバインドするのに使用する。

【0103】バインド要素によって消費される認識結果は、認識結果を指定するためのセマンティックマークアップ言語（SML）を含むXMLドキュメントでよい。その内容は、意味値、話された実際の単語、および信頼度スコアを含む。SMLは、代替の認識選択肢（N番目により認識結果におけるものなど）も含むことができる。発声「I'd like to travel from Seattle to Boston（シアトルからボストンまで行きたい）」に対するSMLドキュメントの例を下に示す。

【0106】・test：任意選択。認識結果を割り当てる際の条件を指示するXML Pattern（W3C XML DOM仕様と同様）ストリング。デフォルト条件は真。

【0107】・value：必須。ターゲット要素に割り当てる認識結果ドキュメントの値を指定するXPath（W3C XML DOM仕様と同様）ストリング。

【0108】例：上記のSMLのリターンを与えられると、以下のreco要素はバインドを使用して、origin_cityおよびdest_city中の値を、ターゲットページの要素textBoxOriginおよびtextBoxDestに転送する。

```

<input name="txtBoxOrigin" type="text"/>
<input name="txtBoxDest" type="text"/>

<reco id="travel">
  <grammar src="/city.xml" />

  <bind      targetElement="txtBoxOrigin"
    value="//origin_city" />
  <bind      targetElement="txtBoxDest"
    value="//dest_city" />
</reco>

```

このバインドは、バインド操作の事前条件として `dest_city` 結果の信頼度属性にテストを行う以下のように条件付きの場合もある。

```

<bind targetElement="txtBoxDest"
  value="//dest_city"
  test="/sml/dest_city[@confidence >= 40]"
/>

```

バインド要素は、ダウンレベルまたはアップレベルのブラウザで認識結果を処理する単純な宣言的手段である。より複雑な処理の場合、アップレベルブラウザによってサポートされる `recoDOM` オブジェクトは、`onReco` イベントハンドラを実装して、プログラマ的なスクリプト分析と認識の戻りの後処理を行えるようにする。

【0109】2.2 属性およびプロパティ

以下の属性はすべてのブラウザでサポートされ、プロパティはアップレベルブラウザによってサポートされる。

【0110】2.2.1 属性

以下の `Reco` の属性は、ダイアログターンのために音声レコグナイザを構成するのに使用する。

【0111】`initialTimeout`：任意選択。認識の開始から音声の検出までのミリ秒単位の時間。この値は認識プラットフォームに渡され、これを越えた場合は、`onSilence` イベントが認識プラットフォームから提供される（2.4.2参照）。指定しない場合、音声プラットフォームはデフォルト値を使用する。

【0112】`babbleTimeout`：任意選択。音声の検出後にレコグナイザが結果を戻さなければならぬミリ秒単位の期間。自動モードおよびシングルモードの `reco` の場合、これは音声検出から `stop` 呼び出しまでの期間に該当する。「複数」モードの `reco` の場合、このタイムアウトは、音声検出から各認識の戻りまでの期間に相当する。すなわち、各結果の戻りまたは他のイベントの後にこの期間を再び開始する。このタイムアウトを超えると、エラーの発生の有無に応じて異なるイベントを投入する。例えば、発声が例外的に長い場合など、レコグナイザがなおオーディオを処理している場合は、ステータスコード13により（2.4.

4参照）`onNoReco` イベントを投入する。ただし、何らかの他の理由でこのタイムアウトを超えた場合はレコグナイザのエラーである可能性がより高くなり、`onTimeout` イベントを投入する。指定しない場合、音声プラットフォームは内部値を使用する。

【0113】`maxTimeout`：任意選択。認識の開始からブラウザに結果を戻すまでのミリ秒単位の期間。これを超えると、ブラウザによって `onTimeout` イベントが投入され、これにより分散環境におけるネットワークまたはレコグナイザの障害に対処（`cater for`）する。「複数」モードの `reco` の場合は、`babbleTimeout` と同様に、各認識の戻りまたは他のイベントの後にこの期間を再度開始する。`maxTimeout` 属性は、`initialTimeout` と `babbleTimeout` の合計よりも大きくするか、または等しくすべきであることに留意されたい。指定しない場合、この値はブラウザのデフォルトになる。

【0114】`endSilence`：任意選択。自動モードの `Reco` の場合、認識結果を戻すまでの、音声があってはならない発話終了後のミリ秒単位の無音期間。自動モード以外のモードの `reco` については無視する。指定しない場合は、プラットフォームの内部値になる。

【0115】`reject`：任意選択。認識拒絶の閾値。これを下回ると、プラットフォームは「`no_reco`」イベントを投入する。指定しない場合、音声プラットフォームはデフォルト値を使用する。信頼度スコアは、0から100の範囲（整数）。拒絶値はこの範囲内にある。

【0116】`server`：任意選択。音声プラットフォームのURI（タグインタープリタと認識プラットフォームをまとめて配置しない場合に使用する）。値の例は、`server=protocol://your speechplatform` などとなる。アプリケーションの作成者（`author`）は、URI スtring にクエリStringを加えることにより、音声プラットフォームに固有の設定を提供することもできる。例：`protocol://yourspeechplatform?bargeinEnergyThreshold=0.5`。

【0117】`langID`：任意選択。音声エンジンが使用する言語を指定するString。String形式は、`xml:lang` 定義に従う。例えば、`lang="en-us"` は米国英語を表す。この属性は、文法要素中で `langID` を指定しない場合のみに有効である（2.1.1参照）

`mode`：任意選択。とるべき認識モードを指定するString。指定しない場合は、「自動」モードになる。

【0118】2.2.2 プロパティ

以下のプロパティは、認識プロセスによって戻される結果を含む（これらはアップレベルブラウザにサポートされる）。

【0119】・`recoResult`：読み取り専用。認識の結果、2.1.2で述べたように、セマンティックマークアップ言語（SML）を含むXML DOMノードオブジェクト中に保持される。認識が行われなかった場合、このプロパティはヌルに戻る。

【0120】・`text`：読み取り／書き込み。認識された単語のテキストを保持するストリング（すなわち、読み取りモードにおける`recoResult`中のSML認識の戻しの中の最上位要素のテキスト属性の内容を表す省略表現）。書き込みモードでは、ストリングを割り当てることができ、次いでそのストリングが認識結果に対応するものとしてそれを構文解析する。書き込みモードでは、このマークアップタグおよびその処理を、クライアントデバイスの他のコンポーネントまたはアプリケーションに拡張することができる。このストリングは、「`smex`」メッセージオブジェクトから得られる。

【0121】・`status`：読み取り専用。認識プラットフォームが返すステータスコード。可能な値は、認識が成功した場合の0、あるいは障害値-1から-4（`Start`メソッド（節2.3.1）および`Activate`メソッド（節2.3.4）で可能な例外で定義する）、およびレコグナイズイベントを受け取った際にセットされるステータス-11から-15（2.4参照）。

【0122】2.3 オブジェクトメソッド

`reco`の起動および文法の起動は、`Reco`のDOMオブジェクト中の以下のメソッドを使用して制御することができる。これらのメソッドにより、アップレベルブラウザは`Reco`オブジェクトの開始および中止、進行中の認識のキャンセル、個々の文法のトップレベルの規則の起動および非活動化を行うことができる（アップレベルブラウザのみ）。

【0123】2.3.1 `Start`

`Start`メソッドは、明示的には非活動化していない認識コンテキストについてのすべての最上位規則をアクティブな文法として使用して認識プロセスを開始する。
構文：`Object.Start()`

戻り値：なし

例外：このメソッドは、非ゼロのステータスコードをセットし、障害があった際は`onNoReco`イベントを発生させる。可能性のある障害には、文法が存在しない（`reco`ステータス=-1）、文法のコンパイルの失敗、存在しないURIなど様々な原因になりうる文法のロードの失敗（`reco`ステータス=-2）、あるいは音声プラットフォームのエラー（`reco`ステータス=-3）などが含まれる。

-3）などが含まれる。

【0124】2.3.2 `Stop`

`Stop`メソッドは、認識プロセスを終了する呼び出しである。`Reco`オブジェクトはオーディオの記録を中止し、レコグナイザは、記録が中止される時点までに受け取ったオーディオについての認識結果を戻す。`Reco`が使用するすべての認識リソースは解放され、その文法は非活動化される。（このメソッドは、自動モードによる通常の認識には明示的に使用する必要がないことに留意されたい。これは、レコグナイザ自体が、完全な文を認識した後のエンドポイント検出において`reco`オブジェクトを中止するからである。）`Reco`が開始されていない場合、この呼び出しは効果を持たない。

構文：`Object.Stop()`

戻り値：なし

例外：なし

【0125】2.3.3 `Cancel`

`Cancel`メソッドは、レコグナイザへのオーディオの供給を中止し、文法を非活動化し、レコグナイザを解放し、すべての認識結果を破棄する。ブラウザは、キャンセルされた認識についての認識結果は破棄する。レコグナイザが開始されていない場合、この呼び出しは効果を持たない。

構文：`Object.Cancel()`

戻り値：なし

例外：なし

【0126】2.3.4 `Activate`

`Activate`メソッドは、文脈自由文法（CFG）の最上位規則を起動する。起動は、「開始された」認識プロセス中には効果を持たないので、認識が開始する前に呼び出さなければならない。明示的に非活動化していない認識コンテキストについてのすべての文法の最上位規則は、すでにアクティブであると見なすことに留意されたい。

構文：`Object.Activate(strName)`

パラメータ：

・`strName`：必須。起動する規則名。

戻り値：なし

例外：なし

【0127】2.3.5 `Deactivate`

このメソッドは、文法中のトップレベル規則を非活動化する。その規則が存在しない場合、このメソッドは効果を持たない。

構文：`Object.Deactivate(strName)`

パラメータ：

・`strName`：必須。非活動化する規則名。空ストリングはすべての規則を非活動化する。

戻り値：なし

例外：なし

【0128】2.4 Reco イベント

Reco DOMオブジェクトは以下のイベントをサポートし、そのハンドラはreco要素の属性として指定することができる。

【0129】2.4.1 onReco：このイベントは、レコグナイザが、そのブラウザで利用することのできる認識結果を得ると起動される。自動モードのreco

の場合、このイベントは認識プロセスを自動的に中止し、リソースをクリアする（2.3.2参照）。onRecoは通例、認識結果のプログラムの分析と、ページ中への結果の処理に使用される。

構文：

【0130】

【表1】

インラインHTML	<Reco onReco="handler">
イベントプロパティ	Object.onReco = handler; Object.onReco = GetRef("handler");

【0131】イベントオブジェクト情報：

【表2】

【0132】

バブル	なし
起動するには	ユーザが何かを言う
デフォルトアクション	認識結果オブジェクトを戻す

【0133】イベントプロパティ：イベントハンドラはプロパティを直接受け取ることはないが、ハンドラはデータについてこのイベントオブジェクトに照会を行うことができる（下記の例のイベントオブジェクトの使用を参照のこと）。

【0134】例

次のXHTMLの断片ではonRecoを使用して、認識結果を構文解析し、その値を適切なフィールドに割り当てるスクリプトを呼び出している。

```
<input name="txtBoxOrigin" type="text" />
<input name="txtBoxDest" type="text" />
<reco onReco="processCityRecognition()" />
  <grammar src="/grammars/cities.xml" />
</reco>
```

```
<script><![CDATA[
    function processCityRecognition () {
        smlResult =
            event.srcElement.recoResult;

        origNode =
            smlResult.selectSingleNode("//origin_city");
        if (origNode != null)
            txtBoxOrigin.value = origNode.text;

        destNode =
            smlResult.selectSingleNode("//dest_city");
        if (destNode != null) txtBoxDest.value
            = destNode.text;

    }
}]></script>
```

【0135】2.4.2 onSilence：onSilenceは、RecoのinitialTimeout属性で指定された時間が過ぎる前に、認識プラットフォームが検出した無音声のイベントに対処する（2.2.1参照）。このイベントは、自動認識モードの認識プロセスを自動的にキャンセルする。

構文：

【0136】

【表3】

インラインHTML	<Reco onSilence="handler" ...>
イベントプロパティ (ECMAScript)	Object.onSilence = handler Object.onSilence = GetRef("handler");

【0137】 イベントオブジェクト情報：
【0138】

【表4】

バブル	なし
起動するには	initialTimeout 属性で指定される期間中にレコグナイザが 音声を検出しなかった
デフォルトアクション	ステータス=-11 にセット

【0139】 イベントプロパティ：イベントハンドラは、プロパティを直接受け取ることはないが、ハンドラはデータについてイベントオブジェクトに照会を行うことができる。

【0140】 2.4.3 onTimeout
onTimeoutは、通例は音声プラットフォームからのエラーを反映する2タイプのイベントを扱う。

【0141】 ・認識が完了する前にmaxTime属性で指定された期間を過ぎた(2.2.1参照)ことを通知する、タグインタープリタが投入するイベントを扱

う。このイベントは通例、分散型アーキテクチャで生じる問題を反映する。

【0142】 ・また、(ii) 認識が開始されたが、bubbleTimeoutで指定された期間内に認識がないまま処理が中止した際に、音声認識プラットフォームが投入するイベントも扱う(2.2.1参照)。このイベントは、認識プロセスを自動的にキャンセルする。

構文：

【0143】

【表5】

インラインHTML	<Reco onTimeout="handler" ...>
イベントプロパティ (ECMAScript)	Object.onTimeout = handler Object.onTimeout = GetRef("handler");

【0144】 イベントオブジェクト情報：
【0145】

【表6】

バブル	なし
起動するには	認識の中止前に、maxTime 属性で設定された期間が過ぎるとブラウザが投入する
デフォルトアクション	reco ステータスを-12にセットする

【0146】 イベントプロパティ：イベントハンドラはプロパティを直接受け取ることはないが、ハンドラはデータについてこのイベントオブジェクトに照会を行うことができる。

【0147】 2.4.4 onNoReco:onNoRecoは、有効な認識結果を戻すことができない際に音声認識プラットフォームが投入するイベント用のハン

ドラである。それが発生しうる異なるケースは、ステータスコードで区別する。このイベントは認識プロセスを自動的に中止する。

構文：

【0148】

【表7】

インラインHTML	<Reco onNoReco="handler">
イベントプロパティ	Object.onNoReco = handler; Object.onNoReco = GetRef("handler");

【0149】 イベントオブジェクト情報：
【0150】

【表8】

バブル	なし
起動するには	レコグナイザが音を検出するが、発声を解釈することができない
デフォルトアクション	ステータスプロパティをセットし、スルの認識結果を戻す。ステータスコードは以下のようにセットする。 ステータス=13:音が検出されたが、解釈できる音声なかった場合 ステータス=14:いくつかの音声が発出され解釈されたが、信頼度が不十分であるために拒絶された場合 (閾値の設定については2.2.1の拒絶の属性を参照のこと) ステータス=15:音声が発出され解釈されたが、音声の検出からbabbleTimeout属性で指定された期間まで完全な認識を戻すことができなかった場合 (2.2.1参照)

【0151】イベントプロパティ: イベントハンドラはプロパティを直接受け取ることはないが、データについてこのイベントオブジェクトに照会を行うことができる。

【0152】3 プロンプト

プロンプト要素は、システム出力を指定するのに使用する。その内容は以下の1つまたは複数にすることができる。

- ・インラインテキストまたは参照テキスト。これは、韻律的な (prosodic) またはその他の音声出力情報でマークアップすることができる。
- ・レンダリング時にそれを含むドキュメントから取り出す変数値。

```
<prompt id="Welcome">
```

ACME天気予報へのお電話ありがとうございます

```
</prompt>
```

この簡単なテキストは、下記に説明する種類のどのマークアップもさらに含むことができる。

【0155】3.1.1 音声合成マークアップ

このプロンプト要素の内部では、どの形式の音声合成マ

```
<prompt id="giveBalance">
```

あなたの口座の残高は<emph>5ドル</emph>です

```
</prompt>
```

【0156】3.1.2 動的な内容

このプロンプトの実際の内容は、プロンプトの出力の直前にクライアントで計算する必要がある場合がある。例えば特定の値を確定するには、ある変数にその値をデリフェレンスする必要がある。この値要素はこの目的に使用することができる。

【0157】値要素

value: 任意選択。ドキュメント中の要素の値を取り出す。

属性:

- ・targetElement: 任意選択。hrefまたはtargetElementを指定しなければなら

す変数値。

- ・オーディオファイルへのリンク。

プロンプト要素は、ダウンレベルブラウザによって宣言的に解釈する (あるいはSMILコマンドで起動すること) も、アップレベルブラウザのオブジェクトメソッドによって宣言的に解釈することもできる。

【0153】3.1 プロンプト内容

プロンプト要素は、テキストまたはオーディオファイルへのリファレンスの形で、あるいはこの両方の形でシステム出力用のリソースを含む。

【0154】簡単なプロンプトは、出力に必要なテキストだけを指定すればよい。例えば、

ークアップ言語でも使用することができる。(この形式は、3.2.1で説明する「tts」属性で指定することができる。) 次の例は、その中の特定の単語を強調する命令を含むテキストを示している。

ない。取り出す値を含む要素のID。

- ・targetAttribute: 任意選択。値を取り出す要素の属性。
- ・href: 任意選択。オーディオセグメントのURL。両方ある場合には、hrefがtargetElementを上書きする。

【0158】targetElement属性は、それを含むドキュメント中の要素を参照するのに使用される。targetElementによってIDが指定された要素の内容を、合成するテキストに挿入する。所望の内容がその要素の属性に保持されている場合、targetAttributeを使用して、targetE

lementの必要な属性を指定することができる。これは、例えば、HTMLフォームコントロール中の値をデリフェレンスするのに有用である。下の例では、「`textBoxOrigin`」要素および「`textBoxDest`」要素の「`value`」属性を、プロンプトの出力前にテキストに挿入している。

```
<prompt id="Confirm">
  あなたが行きたいのは
  <value targetElement:"textBoxOrigin"
targetAttribute:"value" />
  から
  <value targetElement:"textBoxDest"
targetAttribute:"value" />
  ですか？
</prompt>
```

【0159】3. 1. 3 オーディオファイル

```
<prompt>
  ビーッという音がしたらメッセージを録音してください
  <value href="/wav/beep.wav" />
</prompt>
```

【0160】3. 1. 4 参照プロンプト

インラインの内容を指定する代わりに、`src`属性を空要素とともに使用し、URIを介して外部の内容を参照することができる。例えば、

```
<prompt id="Welcome"
  src="/ACMEWeatherPrompts#Welcome" />
```

`src`属性の対象は、インラインプロンプトに指定する上記の内容の任意部分またはすべてを保持することができる。

【0161】3. 2 属性およびプロパティ

このプロンプト要素は、以下の属性（ダウンレベルブラウザ）およびプロパティ（ダウンレベルおよびアップレベルブラウザ）を保持する。

【0162】3. 2. 1 属性

- ・`ttss`：任意選択。テキストから音声への合成用のマークアップ言語タイプ。デフォルトは「SAPI 5」。
- ・`src`：インラインプロンプトを指定する場合は任意選択。参照するプロンプトのURI（3. 1. 4参照）。
- ・`bargein`：任意選択。整数。プロンプトの開始から、人間の聴者が再生を中断できるようになるまでのミリ秒単位の時間。デフォルトは無限、すなわちバージンを許可しない。`bargein=0`にすると、即時のバージンが可能になる。これは、プラットフォームがサポートするどの種のバージンにも該当する。`reco`を開始する時間にどちらを使用可能にするかに応じて、キーワードまたはエネルギーベースのバージン時間をこの方式で構成することができる。
- ・`prefetch`：任意選択。ページをロードする際

この値要素は、合成したプロンプトの代わりに、あるいはその中で再生するあらかじめ記録したオーディオファイルを参照するのに使用することができる。次の例では、プロンプトの最後にビーブ音を鳴らしている。

にプロンプトを直ちに合成して、ブラウザにキャッシュするかどうかを示すブールフラグ。デフォルトは偽。

【0163】3. 2. 2 プロパティ

アップレベルブラウザは、プロンプトのDOMオブジェクト中の以下のプロパティをサポートする。

- ・`bookmark`：読み取り専用。遭遇した最後の合成ブックマークのテキストを記録するストリングオブジェクト。
- ・`status`：読み取り専用。音声プラットフォームから戻されるステータスコード。
- ・`innertext`：読み取り専用。このプロパティはプロンプトのテキストの複写（`transcription`）を提供し、それがシンセサイザに送られる。例えば、あるプロンプトがオーディオウェーブファイルの再生を含む場合、このプロパティはそのプロンプトのテキストバージョン（オーディオウェーブファイルとともに記憶することが多い）を提供し、これはその後、例えばクライアントデバイスで実行するコンポーネントまたはアプリケーションにプロンプトのテキストバージョンを提供することにより、表示するか、またはその他の形で使用することができる。また`innertext`プロパティを使用して、動的コンテンツを含むプロンプトのテキストバージョンも提供することができる。

【0164】3. 3 プロンプトメソッド

プロンプトの再生は、プロンプトのDOMオブジェクト中の以下のメソッドを使用して制御することができる。この方式により、アップレベルブラウザは、プロンプトオブジェクトを開始および停止し、進行中のプロンプトを一時停止および再開し、合成音声のスピードおよび音量を変えることができる。

【0165】3.3.1 Start

プロンプトの再生を開始する。引き数が与えられない限り、このメソッドはオブジェクトの内容を再生する。所与の時間に単一のプロンプトオブジェクトだけが「開始される」と考えられるので、Startを連続して呼び出すとすべての再生が連続的に再生される。

【0166】構文: Object.Start([strText])

パラメータ:

・strText: シンセサイズに送信するテキスト。存在する場合にはこの引き数がオブジェクトの内容を上書きする。

戻り値: なし

例外: サーバがすでにオーディオバッファを開放している場合には、ステータス=-1にセットし、onCompleteイベントを発生させる。

【0167】3.3.2 Pause

オーディオバッファをフラッシュすることなく再生を一時停止する。このメソッドは、再生を一時停止または停止している場合には効果を持たない。

構文: Object.Pause();

戻り値: なし

例外: なし

【0168】3.3.3 Resume

オーディオバッファをフラッシュすることなく再生を再開する。このメソッドは、再生が一時停止状態にない場合は効果を持たない。

構文: Object.Resume();

戻り値: なし

例外: 再開が失敗した際に例外を投入する。

【0169】3.3.4 Stop

```
<html>
<title>プロンプトコントロール</title>
<head>
  <script>
    <!--
      function checkKWBargein() {
        news.change (1.0, 0.5); // 確認中は音量を下げる
        if (keyword.text == "") { // 結果が閾値以下である場合
          news.change (1.0, 2.0); // 音量を元に戻す
          keyword.Start (); // 認識を再開
        } else {
          news.Stop (); // キーワード検出! プロンプトを中止
          // 必要事項を行う
        }
      }
    //
  </script>
  <script for="window" event="onload">
    <!--
```

再生がまだ中止されていない場合に再生を中止し、オーディオバッファをフラッシュする。再生がすでに中止されている場合、このメソッドは単にオーディオバッファをフラッシュする。

構文: Object.Stop();

戻り値: なし

例外: なし

【0170】3.3.5 Change

再生の速度および/または音量を変更する。Changeは再生中に呼び出すことができる。

構文: Object.Change(speed, volume);

パラメータ:

・speed: 必須。変化させる係数。

speed=2.0は、現在の速度を2倍を意味しspeed=0.5は、現在の速度の2分の1にすることを意味し、speed=0は、デフォルト値に戻すことを意味する。

・volume: 必須。変化させる係数。

volume=2.0は、現在の音量を倍にすることを意味し、volume=0.5は、現在の音量を半分にすることを意味し、volume=0は、デフォルト値に戻すことを意味する。

戻り値: なし

例外: なし

【0171】3.3.6 プロンプトコントロールの例次の例は、キーワードバージインの機構をサポートしないプラットフォームに対して、上記のメソッドを使用するプロンプトコントロールをオーサリングする仕組みを示している。

【0172】


```
news.Start (); // keyword.Start ();  
//  
</script>  
</head>  
<body>
```

<prompt id="news" bargein="0">
水曜日の株式市場も、投資家が、来週の連邦準備理事会の会合に先立ち大きな動きにつながる材料を得られなかったことから展開に活気がありませんでした。ハイテク銘柄中心のナスダック総合指数は42.51ポイント下落し、2156.26で取引を終えました。ダウジョーンズ工業平均株価は、午後に入って反騰がなく17.05ポイント下落して10866.46で取引を終えました。
- <!--
</prompt>

```
<reco id="keyword">  
  reject="70"  
  onReco="checkKWBargein()" >  
  <grammar  
    src=http://denali/news bargein grammar.xml />  
  </reco>  
</body>  
</html>
```

【0173】3.4 プロンプトイベント

インラインHTML	<prompt onBookmark="handler" ...>
イベントプロパティ	Object.onBookmark = handler Object.onBookmark = GetRef("handler");

【0176】イベントオブジェクト情報： 【表10】
【0177】

バブル	なし
起動するには	レンダリングしたストリング中のブックマークに遭遇する
デフォルトアクション	ブックマークストリングを戻す

【0178】イベントプロパティ：イベントハンドラはプロパティを直接受け取ることはないが、ハンドラはデータについてこのイベントオブジェクトに照会を行うことができる。
【0179】3.4.2 onBargein：ユーザのバーゲインイベントを検出すると発生する。（例えばエネルギー検出やキーワード認識など、何がバーゲイン

プロンプトDOMオブジェクトは以下のイベントをサポートするが、そのハンドラはプロンプト要素の属性として指定することができる。
【0174】3.4.1 onBookmark
合成ブックマークに遭遇すると発生する。このイベントは再生を一時停止しない。
構文：
【0175】
【表9】

イベントを構成するかの決定はプラットフォームによることに留意されたい。）このイベントハンドラを指定しても、自動的にバーゲイン機能がオンになるわけではない。
構文：
【0180】
【表11】

インラインHTML	<prompt onBargein="handler" ...>
イベントプロパティ	Object.onBargein = handler Object.onBargein = GetRef("handler");

【0181】イベントオブジェクト情報：
【0182】

【表12】

バブル	なし
起動するには	バーgeインイベントに遭遇する
デフォルトアクション	なし

【0183】イベントプロパティ：イベントハンドラはプロパティを直接受け取ることはないが、ハンドラはデータについてこのイベントオブジェクトに照会を行うことができる。

ンプトの再生が最後に達するか、または例外（上記に定義）に遭遇すると発生する。

構文：

【0185】

【0184】3.4.3 onComplete：プロ

【表13】

インラインHTML	<prompt onComplete="handler" ...>
イベントプロパティ	Object.onComplete = handler Object.onComplete = GetRef("handler");

【0186】イベントオブジェクト情報：
【0187】

【表14】

バブル	なし
起動するには	プロンプト再生が完了する
デフォルトアクション	再生が正常に完了した場合はステータス=0 にセットし、 その他の場合は上記に指定するようにステータスをセットする

【0188】イベントプロパティ：イベントハンドラはプロパティを直接受け取ることはないが、ハンドラはデータについてこのイベントオブジェクトに照会を行うことができる。

正か目的地の提供のいずれかであるユーザ応答の意味を判定する仕組みを示している。onBargeinハンドラが、プロンプト中に遭遇した最後のブックマークにグローバルな「mark」変数を設定するスクリプトを呼び出し、この「mark」の値をrecoの後処理関数（「heard」）で使用して、正しい値をセットしている。

【0190】

【0189】3.4.4 ブックマークおよびイベントの使用

次の例は、プロンプトの出力中にバーgeインが行われた場合に、ブックマークイベントを使用して、出発地の訂

```
<script><![CDATA[
    var mark;
    function interrupt( ) {
        mark = event.srcElement.bookmark;
    }
    function ProcessCityConfirm() {
        confirm.stop(); // オーディオバッファをフラッシュする
        if (mark == "mark_origin_city")
            txtBoxOrigin.value =
event.srcElement.text;
        else
            txtBoxDest.value =
event.srcElement.text;
    }
}]></script>
<body>
    <input name="txtBoxOrigin" value="Seattle"
type="text"/>
    <input name="txtBoxDest" type="text" />
```

```

...
<prompt id="confirm" onBargein="interrupt()"
bargein="0">
    <bookmark mark="mark_origin_city" />
    <value targetElement="origin"
targetAttribute="value" />から
    <bookmark mark="mark_dest_city"
/>行きたい行先地を言って下さい
</prompt>
<reco onReco="ProcessCityConfirm()" >
    <grammar src="/grm/1033/cities.xml" />
</reco>
...
</body>

```

【0191】4 DTMF

DTMF認識オブジェクトを作成する。このオブジェクトは、インラインのマークアップ言語構文を使用して、あるいはスクリプト中にインスタンス化することができる。起動すると、DTMFにより、プロンプトオブジェクトがバージンイベントを発生することができる。下記でDTMFとの関連で説明するタグおよびイベントリング、および節5で説明する制御は、一般には、音声ブラウザ216とメディアサーバ214間の対話に関連するものであることに留意されたい。

【0192】4.1 内容

- ・ `dtmfgrammar` : インライン文法
- ・ `bind` : DTMFの変換結果を適切なフィールドに割り当てる

属性:

- ・ `targetElement` : 必須。部分的な認識結果を割り当てる要素 (参照: W3C SMIL 2.0 に同じ)。
- ・ `targetAttribute` : 認識結果を割り当てるターゲット要素の属性 (参照: SMIL 2.0 に同じ)。デフォルトは「`value`」。
- ・ `test` : 割り当ての条件。デフォルトは真。

【0193】例1: テキストにキーをマッピングする

```

<input type="text" name="city"/>
<DTMF id:"city_choice" timeout="2000"
numDigits="1">
    <dtmfgrammar>
        <key value="1">シアトル</key>
        <key value="2">ボストン</key>
    </dtmfgrammar>
    <bind targetElement="city"
targetAttribute:"value" />
</DTMF>

```

「`city_choice`」を起動して、ユーザが1を

```

<input type="text" name="credit_card_number" />
<prompt onBookmark="dtmf.Start(); speech.Start()"

```

押すと「`Seattle`」が入力フィールドに割り当てられ、2を押すと「`Boston`」が割り当てられ、その他の場合は何も割り当てられない。

【0194】例2: どのようにしてDTMFを複数フィールドに使用することができるか

```

<input type="text" name="area_code"/>
<input type="text" name="phone_number" />
<DTMF id:"areacode" numDigits="3"
onReco="extension.Activate()"
    <bind targetElement="area_code" />
</DTMF>
<DTMF id:"extension" numDigits="7">
    <bind targetElement="phone_number" />
</DTMF>

```

この例は、いかにしてユーザが複数フィールドに入力するのを可能にするかを示している。

【0195】例3: 音声入力およびDTMF入力をともに許可し、ユーザがDTMFを開始した際に音声を使用不可にするには

```

        bargein="0">
        <bookmark name="starting" />と言うか、またはあなたのクレジット
        カード番号を入力してください
        </prompt>
        <DTMF id="dtmf" escape="#" length="16"
        interdigitTimeout="2000"
        onkeypress="speech.Stop()">
        <bind targetElement="credit_card_number" />
        </DTMF>
        <reco id="speech" >
        <grammar src="/grm/1033/digits.xml" />
        <bind targetElement="credit_card_number" />
        </reco>

```

【0196】4. 2 属性およびプロパティ

4. 2. 1 属性

・ `dtmfgrammar` : 必須。DTMF文法のURI。

【0197】4. 2. 2 プロパティ

・ `DTMFgrammar` 読み取りおよび書き込み。ストリング変換行列に対するDTMFを表すXML DOM ノードオブジェクト (DTMF文法とも呼ぶ)。デフォルト文法は、

```

<dtmfgrammar>
  <key value="0">0</key>

  <key value="1">1</key>
  ...
  <key value="9">9</key>
  <key value="#">*</key>
  <key value="#">#</key>
</dtmfgrammar>

```

【0198】・ `flush`

読み取り／書き込み。起動の前に、基礎となる電話インタフェースカードのDTMFバッファを自動的にフラッシュするかどうかを示すブールフラグ。デフォルトは偽になり、タイプaheadを使用可能にする。

【0199】・ `escape`

読み取り／書き込み。DTMF読み取りセッションを終了するエスケープキー。エスケープキーはワンキーである。

【0200】・ `numDigits`

読み取り／書き込み。DTMF読み取りセッションを終了させるキーストローク数。エスケープおよび長さの両方を指定した場合は、どちらかの条件を満たすとDTMFセッションが終了される。

【0201】・ `dtmfResult`

読み取り専用ストリング。ユーザが入力したDTMFキーを記憶する。タイプした場合は `escape` が結果に含まれる。

【0202】・ `text`

読み取り専用ストリング。空白で分離されたトークンストリングを記憶し、各トークンはDTMF文法に従って変換する。

【0203】・ `initialTimeout`

読み取り／書き込み。最初のDTMFキーストロークを受け取るまでのミリ秒単位のタイムアウト期間。指定しない場合は、電話プラットフォームの内部設定になる。

【0204】・ `interdigitTimeout` 読み取り／書き込み。次の (`adjacent`) DTMFキーストロークまでのミリ秒単位のタイムアウト期間。指定しない場合は、電話プラットフォームの内部設定になる。

【0205】4. 3 オブジェクトメソッド:

4. 3. 1 `Start`

DTMFの割り込みを可能にし、DTMF読み取りセッションを開始する。

構文: `Object.Start()` ;

戻り値: なし

例外: なし

【0206】4. 3. 2 `Stop`

DTMFを使用不可にする。ただし、ユーザが入力したキーストロークはバッファに残る。

構文: `Object.Stop()` ;

戻り値: なし

例外: なし

【0207】4. 3. 3 `Flush`

DTMFバッファをフラッシュする。`Flush`は、DTMFセッション中には呼び出すことができない。

構文: `Object.Flush()` ;

戻り値: なし

例外: なし

【0208】4. 4 イベント

4. 4. 1 `onkeypress`

DTMFキーを押すと発生する。これは、HTMLコントロールから継承したデフォルトイベントを上書きする。ユーザがエスケープキーを押すと、`onKeyPress`ではなく`onRec`イベントが発生する。構文:

【0209】

【表15】

インラインHTML	<DTMF onkeypress="handler" ...>
イベントプロパティ	Object.onkeypress = handler Object.onkeypress = GetRef("handler");

【0210】 イベントオブジェクト情報：

【表16】

【0211】

バブル	なし
起動するには	タッチトーン電話のキーパッドを押す
デフォルトアクション	押されているキーを戻す

【0212】 イベントプロパティ：イベントハンドラはプロパティを直接受け取ることはないが、ハンドラはデータについてこのイベントオブジェクトに照会を行うことができる。

トは、現在のDTMFオブジェクトを自動的に使用不可にする。

構文：

【0214】

【0213】 4. 4. 2 onReco

【表17】

DTMFセッションを終了すると発生する。このイベン

インラインHTML	<DTMF onReco="handler" ...>
イベントプロパティ	Object.onReco = handler Object.onReco = GetRef("handler");

【0215】 イベントオブジェクト情報：

【表18】

【0216】

バブル	なし
起動するには	ユーザがエスケープキーを押す、またはキーストロークの回数が指定の値を満たす
デフォルトアクション	押されているキーを戻す

【0217】 イベントプロパティ：イベントハンドラはプロパティを直接受け取ることはないが、ハンドラはデータについてこのイベントオブジェクトに照会を行うことができる。

と発生する。このイベントは、認識プロセスを自動的に停止する。

構文：

【0219】

【0218】 4. 4. 3 onTimeout

【表19】

タイムアウトまでに、句の終了イベントを受け取らない

インラインHTML	<DTMF onTimeout="handler" ...>
イベントプロパティ (ECMAScript)	Object.onTimeout = handler Object.onTimeout = GetRef("handler");

【0220】 イベントオブジェクト情報：

【表20】

【0221】

バブル	なし
起動するには	指定のタイムアウト中にDTMF キーストロークが検出されない
デフォルトアクション	なし

【0222】 イベントプロパティ：イベントハンドラはプロパティを直接受け取ることはないが、ハンドラはデ

ータについてこのイベントオブジェクトに照会を行うことができる。

【0223】5 CallControlオブジェクト
電話音声ブラウザの電話インタフェース（呼、端末、および接続）を表す。このオブジェクトは、GUIブラウザ中のwindowオブジェクトと同様にネイティブである。したがって、電話オブジェクトの寿命はブラウザインスタンス自体と同じである。電話用の音声ブラウザは、呼ごとに1つの電話オブジェクトをインスタンス化する。ユーザは、このオブジェクトをインスタンス化または配置しない。

【0224】この点で、このオブジェクトを通じて、ファーストパーティの呼の制御に関連する機能のみを示す。

【0225】5.1 プロパティ

・address

読み取り専用。XML DOM ノードオブジェクト。実装固有。これは発呼者のアドレスである。PSTNの場合は、ANIとALIの組合せにすることができる。VoIPの場合、これは発呼者のIPアドレスになる。

【0226】・ringBeforeAnswer
着信呼に応答するまでの着信音の回数。デフォルトは無限。すなわち、開発者は下記のAnswer（）メソッドを明確に使用して、電話呼に回答しなければならない。コールセンタでACDを使用して着信電話呼をキューに入れる場合、この回数は0にセットしてよい。

【0227】5.2 メソッド

注：ここに示すメソッドはすべて非同期である。

【0228】5.2.1 Transfer

呼を転送する。ブラインド転送の場合、システムは転送が完了すると元の呼を終了し、システムリソースを解放する。

構文：telephone.Transfer(strText)；

パラメータ：

・strText：必須。意図する受信者のアドレス。

戻り値：なし

例外：例えばエンドパーティが話中である、番号が存在しない、ファックスまたは留守番電話が応答するなど、呼の転送が失敗すると例外を投入する。

【0229】5.2.2 Bridge

サードパーティへの転送。呼を転送すると、ブラウザはその呼に割り当てられていたリソースを解放することができる。転送した呼がstrUIDを使用して戻ってきた際にセッション状態を回復するかはアプリケーション次第である。基礎となる電話プラットフォームは、戻ってきた呼を異なるブラウザに経路指定することができる。呼は、受信者がその呼を終了した際のみ戻ることができる。

構文：telephone.Bridge(strText, strUID, [imaxTime])；

パラメータ：

・strText：必須。意図する受信者のアドレス。

・strUID：必須。現在の呼を一意に識別するセッションID。転送した呼が戻される場合、strUIDがアドレス属性に示される。

・imaxTime：任意選択。秒単位の転送呼の最大持続時間。指定しない場合は、プラットフォームの内部値になる。

戻り値：なし

例外：なし

【0230】5.2.3 Answer

電話呼に回答する。

構文：telephone.Answer（）；

戻り値：なし

例外：接続がない際に例外を投入する。この場合onAnswerイベントは発生しない。

【0231】5.2.4 Hangup

電話呼を終了する。その時進行中の呼がない場合は効果を持たない。

構文：telephone.Hangup（）；

戻り値：なし

例外：なし

【0232】5.2.5 Connect

ファーストパーティへのアウトバウンドの電話呼を開始する。

構文：telephone.Connect(strText [iTimeout])；

パラメータ：

・strText：必須。意図する受信者のアドレス。

・iTimeout：任意選択。接続の試みを断念するまでのミリ秒単位の時間。指定しない場合は、プラットフォームの内部値になる。

戻り値：なし

例外：話中音の遭遇、あるいはファックスや留守番電話への到達を含め、呼を完了することができないと例外を投入する（注：ハードウェアがこの機能をサポートしない場合もある）。

【0233】5.2.6 Record

ユーザオーディオをファイルに記録する。

構文：telephone.Record(url, endSilence, [maxTimeout], [initialTimeout])；

パラメータ：

・url：必須。記録された結果のURL。

・endSilence：必須。無音の検出後に記録を中止するミリ秒単位の時間。

・maxTimeout：任意選択。記録を行う秒単位の最大時間。デフォルトはプラットフォーム固有になる。

・initialTimeout：任意選択。記録の開始時に許される無音の最大時間（ミリ秒）。

戻り値：なし

例外：記録をURLに書き込めない際に例外を投入する。

【0234】5.3 イベントハンドラ

電話音声ブラウザを使用するアプリケーション開発者は、以下のイベントハンドラを実装することができる。

【0235】5.3.1 onIncoming ()
音声ブラウザが着信電話呼を受信すると呼び出される。すべての開発者は、電話呼に応答する前にこのハンドラを使用して発呼者のアドレスを読み取り、カスタマイズ

した機能を起動することができる。

【0236】5.3.2 onAnswer ()
音声ブラウザが着信呼に応答すると呼び出される。

【0237】5.3.3 onHangup ()
ユーザが電話を切ると呼び出される。このイベントは、プログラムがHangupメソッドまたはTransferメソッドを呼び出しても自動的に発生しない。

【0238】5.4 例

この例は、電話セッションを操作するために呼制御イベントに結合(wire)したスクリプティングを示す。

```
<HTML>
<HEAD>
  <TITLE>ログオンページ</TITLE>
</HEAD>
<SCRIPT>
  var focus;
  function RunSpeech() {
    if (logon.user.value == "") {
      focus="user";
      p_uid.Start(); g_login.Start();
      dtmf.Start(); return;
    }
    if (logon.pass.value == "") {
      focus="pin";
      p_pin.Start(); g_login.Start();
      dtmf.Start(); return;
    }
    p_thank.Start(); logon.submit();
  }
  function login_reco() {
    res = event.srcElement.recoResult;
    pNode = res.selectSingleNode("//uid");
    if (pNode != null)
      logon.user.value = pNode.xml;
    pNode = res.selectSingleNode("//password");
    if (pNode != null)
      logon.pass.value = pNode.xml;
  }
  function dtmf_reco() {
    res = event.srcElement.dtmfResult;
    if (focus == "user")
      logon.user.value = res;
    else
      logon.pin.value = res;
  }
</SCRIPT>
<SCRIPT for="callControl" event="onIncoming">
  <!--
    // があればアドレスを読み出して、カスタマイズしたものを準備する
    callControl.Answer();
```

```

//
</SCRIPT>
<SCRIPT for="callControl" event="onOffhook">
  <!--
    p_main.Start(); g_login.Start(); dtmf.Start();
    focus="user";
  //
</SCRIPT>
<SCRIPT for="window" event="onload">
  <!--
    if (logon.user.value != "") {
      p_retry.Start();
      logon.user.value = "";
      logon.pass.value = "";
      checkFields();
    }
  //
</SCRIPT>
<BODY>
  <reco id="g_login"
    onReco="login_reco(); runSpeech()"
    timeout="5000"
    onTimeout="p_miss.Start(); RunSpeech()" >
    <grammar
      src=http://kokaneel/etradedemo/speechonly/login.xml/>
    </ reco>
  <dtmf id="dtmf"
    escape="#"
    onkeypress="g_login.Stop();"
    onReco="dtmf_reco();RunSpeech()"
    interdigitTimeout="5000"
    onTimeout="dtmf.Flush();
    p_miss.Start();RunSpeech()" />
  <prompt id="p_main">あなたのユーザIDと個人識別番号を言ってください<
/prompt>
  <prompt id="p_uid">あなたのユーザIDだけを言ってください</prompt>
  <prompt id="p_pin">あなたの個人識別番号だけを言ってください</prompt>
  <prompt id="p_miss">申し訳ありませんが、聞き取れませんでした。</promp
t>
  <prompt id="p_thank">ありがとうございます。あなたの識別を確認する間お
待ちください</prompt>
  <prompt id="p_retry">申し訳ありませんが、あなたのユーザIDと個人識別
番号が一致しません</prompt>
  <H2>Login</H2>
  <form id="logon">
    UID: <input name="user" type="text"
      onChange="runSpeech()" />
    PIN: <input name="pass" type="password"
      onChange="RunSpeech()" />
  </form>

```


</BODY>

</HTML>

【0239】6 ダイアログフローの制御

6.1 HTMLおよびスクリプトを使用してダイアログフローを実装する

次の例は、入力ボックスの値を探して、入力に対して状況依存型のヘルプを提供する単純なダイアログフローの

実装方法を示している。これは、HTML入力機構のタイトル属性（視覚ブラウザで「ツールチップ」機構として使用される）を使用して、ヘルププロンプトの内容を形成するのを補助する。

```
<html>
  <title>状況感知型ヘルプ</title>
<head>
  <script>    var focus;
              function RunSpeech() {
                if (trade.stock.value == "") {
                  focus="trade.stock";
                  p_stock.Start();
                  return;
                }
                if (trade.op.value == "") {
                  focus="trade.op";
                  p_op.Start();
                  return;
                }
                //.. repeat above for all fields
                trade.submit();
              }
              function handle() {
                res = event.srcElement.recoResult;
                if (res.text == "help") {
                  text = "～だけをお願いします";
                  text += document.all[focus].title;
                  p_help.Start(text);
                } else {
                  // proceed with value assignments
                }
              }
  </script>
</head>
<body>
  <prompt id="p_help" onComplete="checkFiledS()" />
  <prompt id="p_stock"
    onComplete="g_stock.Start()">株式銘柄を教えてください</prompt>
  <prompt id="p_op" onComplete="g_op.Start()">売りまたは買いのどちら
    をご希望ですか</prompt>
  <prompt id="p_quantity"
    onComplete="g_quantity.Start()">株式数はいくつですか</prompt>
  <prompt id="p_price"
    onComplete="g_price.Start()">価格はいくらですか</prompt>
  <reco id="g_stock" onReco="handle(); checkFields()" >
    <grammar src="./g_stock.xml" />
  </ reco >
```

```

<reco id="g_op" onReco="handle(); checkFields()" />
  <grammar src="./g_op.xml" />
</ reco >
<reco id="g_quantity" onReco="handle(); checkFields()"
/>
  <grammar src="./g_quant.xml" />
</ reco >
<reco id="g_price" onReco="handle(); checkFields()" />
  <grammar src="./g_quant.xml" />
</ reco >

<form id="trade">
  <input name="stock" title="stock name" />
  <select name="op" title="buy or sell">
    <option value="buy" />
    <option value="sell" />
  </select>
  <input name="quantity" title="number of shares"
/>
  <input name="price" title="price" />
</form>
</body>
</html>

```

【0240】6.2 SMILを使用する eco要素の起動を示す。
 次の例は、SMIL機構を使用したプロンプトおよびr

```

<html xmlns:t="urn:schemas-microsoft-com:time"
      xmlns:sp="urn:schemas-microsoft-com:speech">
<head>
<style>
  .time { behavior: url(#default#time2); }
</style>
</head>
<body>
<input name="txtBoxOrigin" type="text"/>
<input name="txtBoxDest" type="text" />

<sp:prompt class="time" t:begin="0">
  出発地と行先地を教えてください
</sp:prompt>
<t:par t:begin="time.end"
t:repeatCount="indefinitely"
  <sp:reco class="time" >
    <grammar src="./city.xml" />
    <bind targetElement="txtBoxOrigin"
      value="//origin_city" />
    <bind targetElement="txtBoxDest"
      test="/sml/dest_city[@confidence $gt$ 40]"
      value="//dest_city" />
  </sp:reco>

```

```

</t:par>
</body>
</html>

```

【0241】7. SMEX (メッセージ) 要素/オブジェクト

SMEXは、Simple Messaging Exchange/Extensionの略語であるが、これは、クライアントデバイスのプラットフォーム上の外部コンポーネントまたはアプリケーションと通信するオブジェクトである。これは、タグ名<smex>を有する要素として、XMLまたはそれに類似のマークアップベースのドキュメント中に埋め込むことができる。このメッセージングオブジェクトの使用例には、ロギングおよび電話制御を含むことができる。このオブジェクトは、メッセージングを通じて新しい機能を追加することを可能にすることから、マークアップベースの認識およびプロンプティング (prompting) の拡張性を表す。

【0242】インスタンスを生成すると、このオブジェクトは、その構成パラメータまたは属性指定を通じて、プラットフォームコンポーネントまたはアプリケーションとの非同期のメッセージ交換経路を確立するように指示を受ける。このオブジェクトはストリングプロパティを有し、そのプロパティが割り当て動作 (すなわち `value`) を受ける対象である場合には、必ずその内容がプラットフォームコンポーネントまたはアプリケーションに送られる。同様に、このオブジェクトは、プラットフォームコンポーネントまたはアプリケーションから受け取ったメッセージを保持する、XML DOMノードタイプのプロパティも有する。このメッセージオブジェクトは、プラットフォームメッセージを受け取ると必ずイベントを送る。このオブジェクトは、その基本動作

例1: ロギングオブジェクトとしての `smex` の使用

```

<smex_id="logServer">
  <param name="d:server"
xmlns:d="urn:Microsoft.com/COM">
    <d:protocol>DCOM</d:protocol>
    <d:clsid>2093093029302029320942098432098</d:clsid>
    <d:iid>0903859304903498530985309094803</d:iid>
  </param>
</smex>
<listen...>
  ...//rec結果を入力フィールドにバインドする他のディレクティブ
  ..... <bind targetElement="logServer"
targetAttribute="sent"
value="*[@log SgeS 3]/>
</listen>

```

【0246】この例は、COMオブジェクトをそのクラスIDおよびインタフェースIDとともに使用して、ロギング機構を実現する仕組みを示している。音声開発者

が非同期なので、アプリケーション開発者がタイムアウト設定を操作するための内蔵クロックも有する。

【0243】メッセージまたは `smex` オブジェクトは、通信手段にとってアグノスティック (agnostic) である。しかし、一実施形態では、`smex` オブジェクトは、通常のXMLやマークアップ要素と同じ寿命を有する。すなわち、`smex` オブジェクトは、それをホストするドキュメントをアンロードすると消滅する。多くのケースでは、`smex` オブジェクトはアンロードされると自動クリーンアップを実行し、通信リソースを解放することができるが、マークアップページ間で永続的な通信リンクが望ましい使用事例 (例えば呼の制御など) もありうる。そのような事例のために、このアーキテクチャでは、割り振られたリソースを解放する (例えばソケットを閉じるなど) 責任をアプリケーション開発者に課す。

【0244】`smex` オブジェクトは、メッセージのフォーマット (スキーマ) についてはニュートラルである。実施形態によっては、既存の標準的なメッセージフォーマット (例えばSTPまたはCCXMLで使用するものなど) をまず優先して、実装者がいくつかの基本的スキーマをサポートすることを必要とするのが望ましい場合もある。基本的に、このアーキテクチャは、プラットフォーム開発者およびアプリケーション開発者の両者が、XMLあるいはそれに類似のマークアップの規格化された拡張性を最大限に活用して、一方では相互操作性を失うことなく他の機能を導入することを可能にする。

【0245】

は、関連するSMLノードにロギングするための当該レベルを示す属性「log」を付加する。上の例では、アプリケーション開発者が、単一のバインドディレクティ

ブを使用することにより、3を超えるか、または3に等しいログ値を有するノードすべてにログインすることを選択している。この例は、ダウンレベルブラウザでもアップレベルブラウザでも機能する。

【0247】この例はまた、smexオブジェクトがプラットフォームメッセージを認識ドキュメントに伝達する役割を負うような混乱状態がない限り、あるページが、同じプラットフォームコンポーネントと通信する複

例2：着信呼のアドレスの読み取り

```
<input type="text" id="remote"/>
<input type="text" id="transfer"/>
<input type="text" id="local"/>
<input type="hidden" id="session_id"/>
.....
<smex id="telephone" sent="start_listening">
  <param name="server">http://tel-svr/whatever</param>
  <bind targetElement="session_id" value="//sid"/>
  <bind targetElement="remote"
value="//remote_addr"/>
  <bind targetElement="transfer"
value="//transfer_addr"/>
  <bind targetElement="local"
value="//local_addr"/>
  ....
</smex>
```

【0249】この例は、どのようにバインドディレクティブを使用して、受信メッセージを処理できるかを示している。この例では、着信呼のメッセージが、下位要素のremote_addr、transfer_addr、およびlocal_addrを有するものと想定しており、その内容はそれぞれ着信呼のリモートアドレス、転送アドレス、およびローカルアドレスを表す。

【0250】この例では、HTTPに基づくコネクションレスプログラミングを使用して電話サーバと通信する。この場合の電話サーバは、複数のブラウザインスタンスと通信するように設計されており、したがって、各クライアントは、アプリケーションの開始時にサーバから割り当てられる一意のIDによって自らを識別しなければならない。この例では、これはサーバに「start_listening」メッセージを送信することによって実現する。この例では、セッションIDを隠しフィールドに記憶し、それをウェブサーバに送信して、アプリケーションの次のページに渡すことができるが、セッション状態の管理には他の技術（例えばクライアントサイドのクッキー）も使用することができる。recoの場合と同様に、あらゆるプラットフォームメッセージについてすべてのバインドディレクティブが実行されるとは限らない。上の例は、着信電話呼がある際に一意のIDのみを受信することは示唆していない。

【0251】7.1 プロパティ

数のsmexオブジェクトを含むことが可能であることも示すものである。上の例は、あるコンポーネントが複数のインタフェースを実装することができ、それぞれのインタフェースがそれ自体のsmexまたはメッセージ経路を有することを示唆している。これと同じ論議は、複数のポートをリッスンするTCPサーバにも当てはまる。

【0248】

smexオブジェクトは以下のプロパティを有することができるが、初期値指定のための属性としても機能することができるのは、読み取り／書き込みのプロパティだけである。

【0252】・sent：読み取り／書き込み。プラットフォームコンポーネントに送信するメッセージに対応するストリング。lvalueとしてsentを使用する場合は、必ずその内容をディスパッチする。このプロパティをrvalueとして使用する場合、あるいはこのプロパティにヌルオブジェクトを割り当てる場合には効果がない。

【0253】・received：読み取り専用。受信メッセージを表すXML DOMノードデータ。このメッセージは、次のonReceiveイベントが送ることのできる状態になるまで、rvalueとして使用することができる。

【0254】・timer：読み取り／書き込み。タイムアウトイベントをトリガするまでの時間を表すミリ秒単位の数。クロックは、このプロパティに正の値が割り当てられると刻時を開始する。この値は、カウントダウンの進行中に変更することができる。ゼロまたは負の値にすると、タイムアウトイベントをトリガせずにクロックを停止する。デフォルトは0、すなわちタイムアウトなしである。

【0255】・status：読み取り専用。オブジェ

クトの最近のステータスを表す整数。可能な値は、0、-1、および-2であり、それぞれ、正常、タイムアウトの終了、およびプラットフォームとの通信を確立できない、あるいは通信の中断を意味する。受信されるプロパティを通じて、プラットフォーム固有のエラーメッセージを伝達するとよい。エラーメッセージの伝達が成功した場合、ステータスコードは0になる。

【0256】7. 2 イベント

このオブジェクトは以下のイベントを有する。

・onReceive: このイベントは、プラットフォームメッセージが到着すると送られる。バインド要素によって宣言されたディレクティブがある場合には、このイベントを発生させる前にそのディレクティブを先に評価する。イベントを送る前に、受け取ったプロパティを更新する。

【0257】・onError: このイベントは、タイムアウトが経過したとき、あるいは通信リンクエラーに遭遇したときに送られる。このイベントを送る際、上記のように、ステータスプロパティをそれに対応するエラーコードによって更新する。

【0258】7. 3 子要素

ある要素の形を仮定するとき、smexは以下の子要素を有することができる。

・bind: ディレクティブを受信メッセージに作用させる点を除いては、recoの場合と同様。

```
<smex id = "logServer" onload="addFunction( )">
</smex>
<script>
function my_logMessage(logClass, message) {
logServer. sent = logClass + "|" + message;
}
function addFunction( ){
logServer. prototype. logMessage=
my_logMessage;
}
</script>
```

よりオブジェクト指向的な方式でこの関数を参照することができる。logServer. logMessage(RECO_LOG_ERROR, "My message");上記の例のように拡張を機能させるために、smexオブジェクトの実装者にはより多くの作業が要求されるが、すべての必要な機構はすでに確立された規格であることに留意されたい。

【0261】

【発明の効果】以上、説明したように、本発明によれば、インターネットなどのサーバ/クライアントアーキテクチャで入力認識を提供するのに使用されるウェブ対応認識アーキテクチャを有するシステムは、統一したアーキテクチャを備えることが可能となる。

【図面の簡単な説明】

【図1】本発明の実施形態の、コンピューティングデバ

・param: recoの場合と同様。smexオブジェクトのプラットフォーム固有パラメータを提供する。各param要素は、「name」属性を使用して名前をつけることができ、param要素の内容がそのパラメータの値になる。一実施形態では、この要素は、ネームスペースの標準的なXML属性とXMLデータタイプ宣言を理解しているべきである。

【0259】7. 4 その他の補足説明

ロギング機能のためにSMEXを拡張する簡潔な方法の1つが以下である。

```
<smex id="logServer"...>... </smex>
<script> function logMessage(logClass, message){
logServer. sent = logClass + "|" + message;
} </script>
```

これは、実際に、その振る舞いを個別設定することのできる(グローバル)関数でこのオブジェクトを拡張している。上の例では、IDとメッセージの間にフィールド区切り文字「|」を挿入するようにロギング関数をプログラムしている。

【0260】グローバル関数を好まない者は、ECMAScriptの「prototype」プロパティを使用して、この関数をオブジェクトメソッドとして付加することができる。例えば、

イスの動作環境の第1の実施形態の平面図である。

【図2】本発明の実施形態の、図1のコンピューティングデバイスのブロック図である。

【図3】本発明の実施形態の、電話機の平面図である。

【図4】本発明の実施形態の、汎用コンピュータのブロック図である。

【図5】本発明の実施形態の、クライアント/サーバシステムのアーキテクチャのブロック図である。

【図6】本発明の実施形態の、クレジットカード情報を得るための表示の図である。

【図7】本発明の実施形態の、クライアントで実行することのできるマークアップ言語のページの図である。

【図8】本発明の実施形態の、ディスプレイおよび音声認識機能を有するクライアントで実行することのできる

マークアップ言語の例示的ページの図である。

【図9】本発明の実施形態の、音声レンダリングのみを用い、システム主導型で、クライアントで実行できるマークアップ言語の例示的ページの図である。

【図10】本発明の実施形態の、音声レンダリングのみを用い、システム主導型で、クライアントで実行できるマークアップ言語の例示的ページの図である。

【図11】本発明の実施形態の、音声レンダリングのみを用い、混合主導型で、クライアントで実行できるマークアップ言語の例示的ページの図である。

【図12】本発明の実施形態の、音声レンダリングのみを用い、混合主導型で、クライアントで実行できるマークアップ言語の例示的ページの図である。

【図13】本発明の実施形態の、サーバサイドのプラグインモジュールによって実行することのできる例示的スクリプトの図である。

【図14】本発明の実施形態の、認識サーバの第1の動作モードを図式的に示す図である。

【図15】本発明の実施形態の、認識サーバの第2の動作モードを図式的に示す図である。

【図16】本発明の実施形態の、認識サーバの第3の動作モードを図式的に示す図である。

【図17】本発明の実施形態の、スクリプティングを用いないクライアントで実行することのできる宣言的マークアップ言語の例示的ページの図である。

【図18】本発明の実施形態の、スクリプティングを用いないクライアントで実行することのできる宣言的マークアップ言語の例示的ページの図である。

【符号の説明】

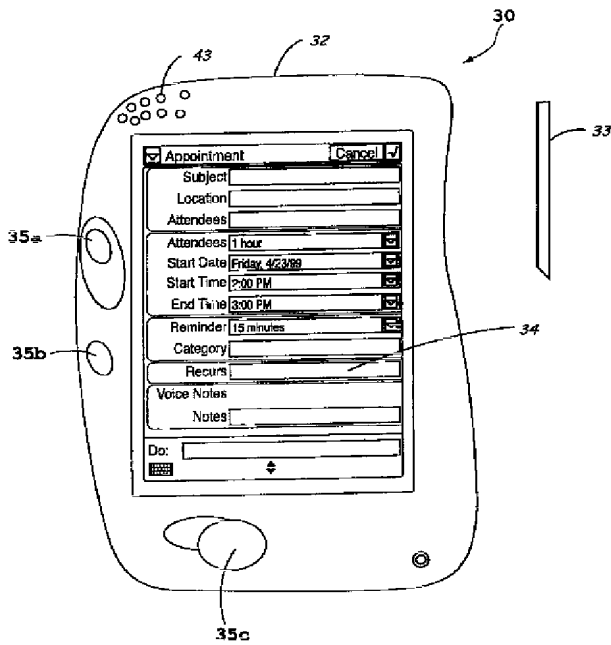
29、183 マイクロフォン
30 データ管理デバイス（モバイルデバイス、クライアント）
32 筐体
33 スタイラス
34 ディスプレイ
35a、35b、35c ボタン
36 キーパッド
37、59 A/D変換器
43、187 スピーカ
50 CPU
52 無線トランシーバ
54、152 RAM
58、151 ROM
60 通信インタフェース
80 電話機
82 ディスプレイ
84 キーパッド
120 汎用コンピュータ
140 プロセッサ
141 システムバス

150 システムメモリ
153 BIOS
154、164 オペレーティングシステム
155、165 アプリケーションプログラム
156、166 プログラムモジュール
157、167 プログラムデータ
161 ハードディスクドライブ
160、170 インタフェース
160 取外し不能不揮発性メモリインタフェース
170 リムーバブル不揮発性メモリインタフェース
171 磁気ディスクドライブ
172 磁気ディスク
175 光ディスクドライブ
176 光ディスク
180 ユーザ入力インタフェース
181 ポインティングデバイス
182 キーボード
184 モニタ
185 ビデオインタフェース
186 プリンタ
188 出力周辺インタフェース
190 ネットワークインタフェース
191 LAN
192 モデム
193 WAN
194 リモートコンピュータ
195 リモートアプリケーションプログラム
200 アーキテクチャ
202 ウェブサーバ
204 認識サーバ
205 ネットワーク
207 専用回線
208 電話網
210 ゲートウェイ
211 レコグナイザ
212 電話音声ブラウザ
214 メディアサーバ
216 音声ブラウザ
250、252、254 フィールド
260、270、300 本体部分
262、272、302 スクリプト部分
264 提出ボタン
280、282、284、405 コード部分
281 スケジュール
283、305 認識の開始
285 音声の検出
287 音声の終了
289、291、293、295、297、299、301、303、305 属性（期間、イベント）
290、303 コントロール

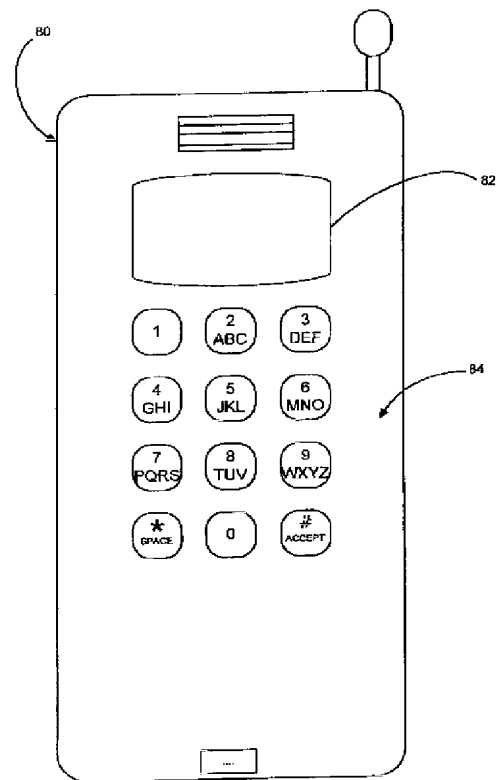
307 音声変換システム
309 パーサ

320、324 モジュール

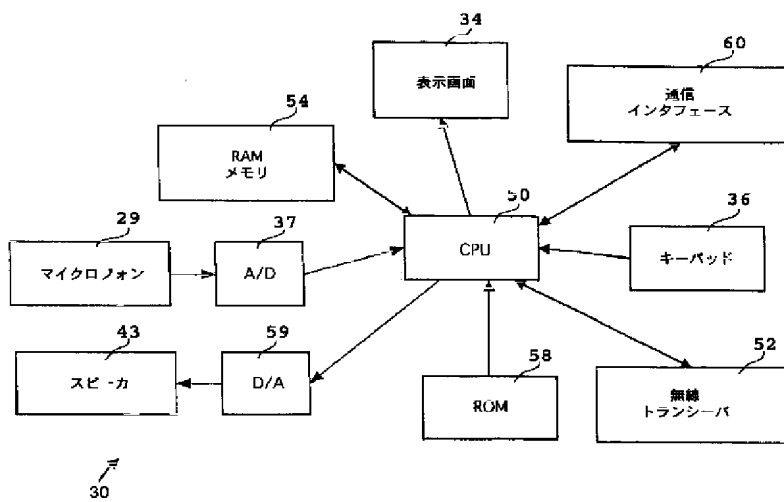
【図1】



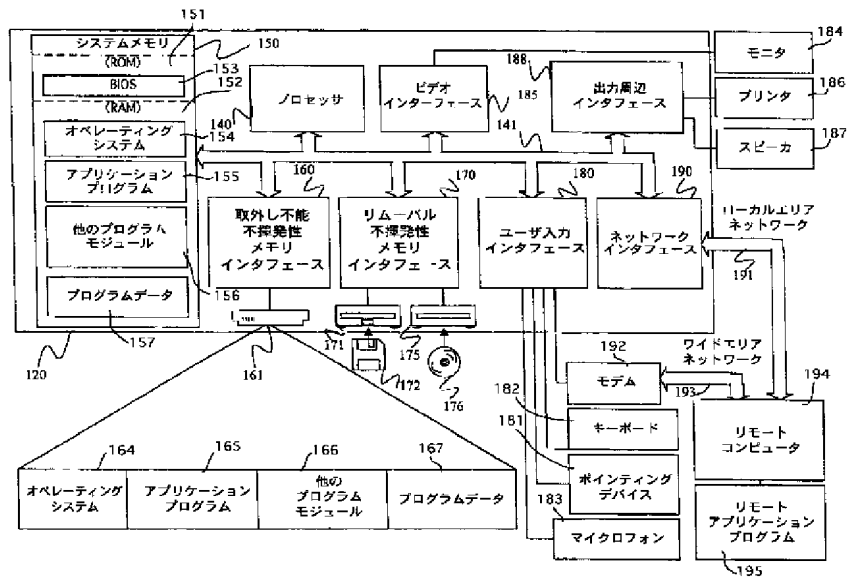
【図3】



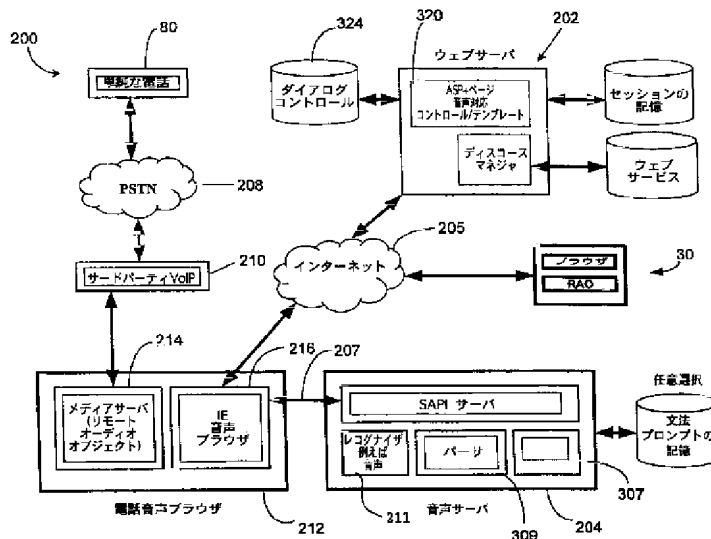
【図2】



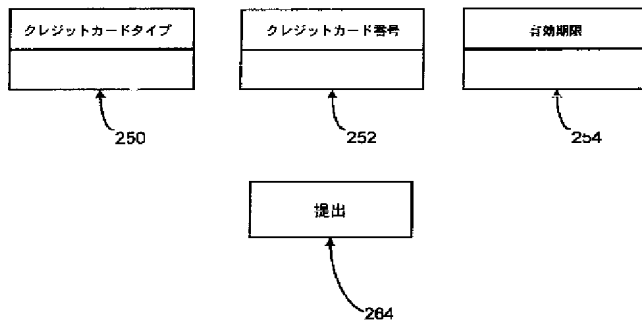
【図4】



【図5】



【図6】



【図7】

```

260  <html>
      <form id="get_card_info" method="post" action="http://payment.asp">
        <select name="card_type">
          <option value="amex">American Express</option>
          <option value="visa">Visa</option>
          <option value="ms">MasterCard</option>
        </select>
        <input type="text" name="card_num" width="30"
          onChange="handle()" />
        <input type="text" name="expiry_date" />
        <input type="submit" value="Submit" onClick="verify()" />
      </form>
      <script><![CDATA[
        function handle() {
          if (get_card_info.card_type.value == "amex") {
            if (get_card_info.card_num.length != 15) {
              alert ("amex should have 15 digits");
            } else
              if (get_card_info.card_num.length != 16)
                alert ("visa and master should have 16 digits");
          }
          function verify() {
            var flag = window.confirm("submit the credit card info?");
            if (flag)
              get_card_info.submit();
          }
        }
      ]]>
    </script>
  </html>
262

```

【図8】

```

270 <html>
      <form id="get_card_info" method="post" action="http://payment.asp">
        <select name="card_type" onClick="talk(g_card_types)">
          <option value="amex">American Express</option>
          <option value="visa">Visa</option>
          <option value="ms">MasterCard</option>
        </select>
        <reco id="g_card_types" onReco = "handle()" >
          <grammar src = "/gram#card_types" />
        </reco>
        <input type="text" name="card_num" width="30"
          onClick="talk(g_card_num)" />
        <reco id="g_card_num" onReco = "handle()" >
          <grammar src = "/gram#digits" />
        </reco>
        <input type="text" name="expiry_date"
          onClick="talk(g_expiry_date)" />
        <reco id="g_expiry_date" >
          <grammar src = "/gram#dates" />
        </reco>
        <input type="submit" value="Submit" onClick="verify()" />
      </form>
      <script><![CDATA[
        function talk(gobj) {
          gobj.activate();
        }
        function handle() {
          if (get_card_info.card_num != null) {
            if (get_card_info.card_type.value == "amex") {
              if (get_card_info.card_num.length != 15)
                alert ("amex should have 15 digits");
            } else
              if (get_card_info.card_num.length != 16)
                alert ("visa and master should have 16 digits");
          }
        }
        function verify() {
          var flag = window.confirm("submit the credit card
info?");
          if (flag) {
            get_card_info.submit();
          }
        }
      ]]>
    </script>
  </html>

```

270 points to the start of the HTML document.

280 points to the first select element.

282 points to the first text input element.

284 points to the second text input element.

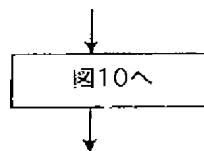
272 points to the start of the JavaScript code block.

【図9】

```

< body >
  < form id = " get_card_info " method = " post " action = " http://payment.asp "
    onActivate = " welcome() ">
    < prompt id = " p_welcome "> We now need your credit card </ prompt >
    < prompt id = " p_mumble "> I didn't understand you </ prompt >
    < prompt id = " p_card_type " bargain = " true "> What credit card would you
300 use? </ prompt >
    < prompt id = " p_card_num " bargain = " true "> Please say the number </ prompt >
    < prompt id = " p_expiry_date " bargain = " true "> What is the expiration
303 date? </ prompt >
    < prompt id = " p_content ">
      I have your < value select = " card_type " /> < value select = " card_num " />
      with expiration date
      < value select = " expiry_date " />
    </ prompt >
    < prompt id = " p_confirm "> Is this correct? </ prompt >
    < reco id = " g_card_types " onNoReco = " mumble(this, 2) ">
      onReco = " _handle(this, card_type) " />
    < grammar src = " ./gram#card_types "
    </ reco >
    < reco id = " g_card_num " onNoReco = " mumble(this, 1) ">
      onReco = " _handle(this, card_num) " />
    < grammar src = " ./gram#digits " />
    </ reco >
    < reco id = " g_expiry_date " onNoReco = " mumble(this, 1) ">
305 onReco = " _handle(this, expiry_date) " audio = " rao " />
    < grammar src = " ./gram#dates " />
    </ reco >
    < reco id = " confirmation " onReco = " confirmed(this) " />
    < grammar src = " ./gram#yesno " />
    </ reco >
    < select name = " card_type ">
      < option value = " amex "> American Express </ option >
      < option value = " visa "> Visa </ option >
      < option value = " ms "> MasterCard </ option >
    </ select >
    < input type = " text " name = " card_num " width = " 30 " />
    < input type = " text " name = " expiry_date " />
    < input type = " submit " value = " Submit " />
  </ form >
  < script > <![CDATA[
    function welcome() {
      p_welcome.active();
      repeat = 0;
      checkFilled();
    }
    function mumble(gobj, maxprompts) {
      gobj.deactivate();
      p_mumble.active();
      checkFilled();
    }
  ]]>

```



【図11】

```

<body>
  <form id="get_card_info" method="post" action="http://payment.asp"
    onactivate="welcome()">
    <prompt id="p_welcome">We now need your credit card</prompt>
    <prompt id="p_mumble">I didn't understand you</prompt>
    <prompt id="p_card_type" bargain="true">What credit card would you
use?</prompt>
    <prompt id="p_card_num" bargain="true">Please say the number</prompt>
    <prompt id="p_expiry_date" bargain="true">What is the expiration
date?</prompt>
    <prompt id="p_content">
      I have your <value select="card_type" /> <value select="card_num" />
with expiration date <value select="expiry_date" />
    </prompt>
    <prompt id="p_confirm">Is this correct?</prompt>
    <reco id="g_get_card_info" onReco="_handle()" onNoReco="mumble(this)">
      <grammar src="./gram/getPayment" />
      <bind target="card_type" value="/card/type" />
      <bind target="card_num" value="/card/number" />
405 — <bind target="expiry_date" value="/card/expr_date" />
    </reco>
    <reco id="confirmation" onReco="confirmed(this) onNoReco="mumble(this)" />
      <grammar src="./gram/yesno" />
    </reco>
    <select name="card_type">
      <option value="amex">American Express</option>
      <option value="visa">Visa</option>
      <option value="ms">MasterCard</option>
    </select>
    <input type="text" name="card_num" width="30" />
    <input type="text" name="expiry_date" />
    <input type="submit" value="Submit" />
  </form>
  <script><![CDATA[
    function welcome() {
      p_welcome.active();
      repeat = 0;
      checkFilled();
    }
    function mumble(gobj) {
      gobj.deactivate();
      p_mumble.active();
      checkFilled();
    }
    function _handle() {
      handle();
      checkFilled();
    }
  ]]>

```

↓

図12へ

↓

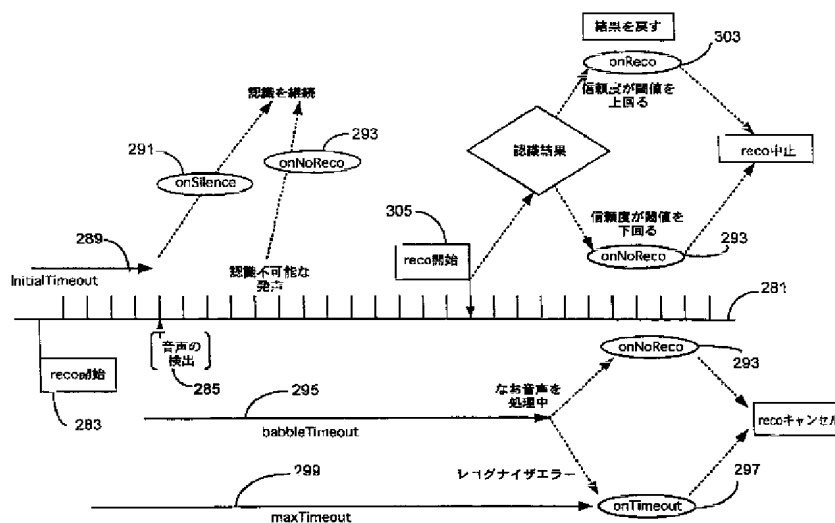
【図12】

```

function checkFilled() {
  if (card_type.value == "") {
    p_card_type.activate(); do_field.activate(); return;
  }
  if (card_num.value == "") {
    p_card_num.activate(); do_field.activate(); return;
  }
  if (expiry_date.value == "") {
    p_expiry_data.activate(); do_field.activate(); return;
  }
  p_content.activate();
  p_confirm.activate();
  confirmation.activate();
}
function confirmed(gobj) {
  if (gobj.recogRes.text == "yes")
    get_card_info.submit(gensml());
}
//
// user codes start here
//
function handle() {
  if (field == get_card_info.card_num) {
    if (get_card_info.card_num.length != 15) {
      prompt.speak ("amex should have 15 digits");
      get_card_info.card_num = "";
    }
  } else
    if (get_card_info.card_num.length != 16) {
      prompt.speak ("visa should have 15 digits");
      get_card_info.card_num = "";
    }
}
function gensml() {
  str = '<sm1><credit_card type="';
  str += card_type.value; str += '"><number>';
  str += card_number.value; str += '</number><expire>';
  str += expiry_date.value; str +=
  '</expire></credit_card></sm1>';
  return str;
}
}]>
</script>
</body>

```

【図15】



【図13】

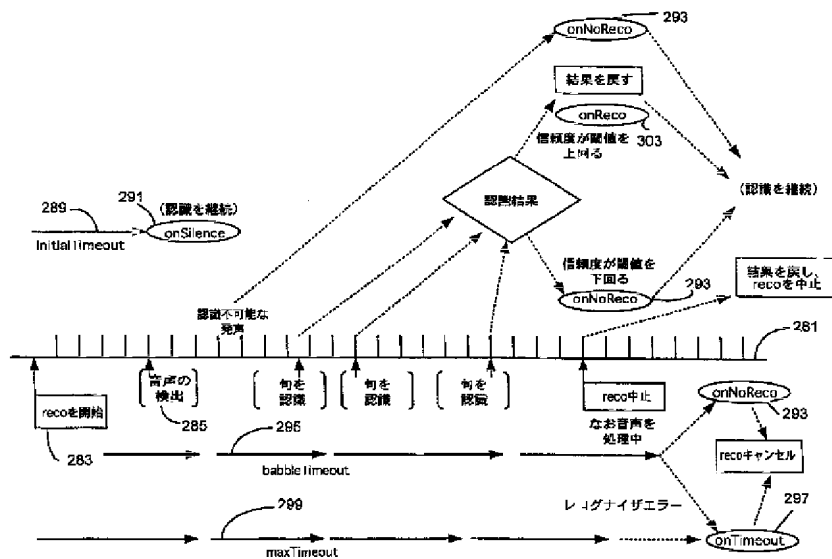
ASP+ページの一例

```

<%@ Page language="Jscript" AutoEventWireup="false" Inherits="Credit.Transaction" %>
<html><head>
<!-- ASPX page for both voice only & multimodal credit card example -->
<script>
function handle() {
    if (field == get_card_info.card_num) {
        if (get_card_info.card_num.length != 15) {
            prompt.speak ("amex should have 15 digits");
            get_card_info.card_num = "";
        }
    } else
        if (get_card_info.card_num.length != 16) {
            prompt.speak ("visa should have 15 digits");
            get_card_info.card_num = "";
        }
    }
function gensml() {
    str = '<sml><credit_card><card_type>';
    str += card_type.value; str += '</card_type><number>';
    str += card_number.value; str += '</number><expire>';
    str += expiry_date.value; str += '</expire></credit_card></sml>';
    return str;
}
</script>
<script runat="server">
function Page_Load (obj, args) {
    if (PostBack) {
        validator = new System.Speech.SMLValidator("./CreditSDL.xml");
        dsml = validator.Evaluate(args);
        Navigate(ChoosePage(dsml));
    } else {
        // initialize fields with args
    }
}
</script>
</head>
<body>
<speech:form id="get_card_info" style="system initiative"
    prompt="./prompt/getPayment" onsubmit="gensml()">
    <speech:choice name="card_type" prompt="What credit card would you use?"
        grammar="./gram#card_types" onPhraseFinish="handle()">
        <option>American Express</option>
        <option>Visa</option>
        <option>Mastercard </option>
    </speech:choice>
    <speech:textbox name="card_number" prompt="Please say the number"
        grammar="./gram#digits" onPhraseFinish="handle()">
    <speech:textbox name="expiry_date" prompt="What is the expiration date? "
        grammar="./gram#dates" onPhraseFinish="handle()"/>
</speech:form>
</body>
</html>

```

【図16】



【図17】

```

<html>
<body>
  <!-- the data section -->
  <form id="get_drink">
    <input name="drink" />
    <input type="radio" name="cream"/>
    <input type="radio" name="sugar"/>
    <input name="uid" type="hidden"/>
  </form>
  <!-- The speech section -->
  <prompt id="welcome">Welcome, caller! </prompt>
  <prompt id="ask">Do you want coke, coffee, or orange juice?
  </prompt>
  <prompt id="confirm">I heard <value href="drink"/>. Is this
  correct? </prompt>
  <prompt id="thanks">Thank you. Please wait when I get it for
  you. </prompt>
  <prompt id="retry">Okay, let's do this again </prompt>
  <prompt id="reprompt">Sorry, I missed that. </prompt>
  <prompt id="cream_sugar">Do you want cream or sugar with your
  coffee?</prompt>

  <reco id="reco_drink"><grammar src="./drinktypes"/>
  354 364 <bind test="//[@confidence $!e$ 10]"
    targetElement="reprompt" targetMethod="start"
  366 targetElement="ask" targetMethod="start"
  368 targetElement="reco_drink" targetMethod="start"/>
  370 <bind test="//drink/coffee[@confidence $!e$ 10]"
    targetElement="drink" value="//drink"
  372 targetElement="cream_sugar" targetMethod="start"
  374 targetElement="reco cream sugar"
  376 targetMethod="start"/>
    <bind test="//[@confidence $!e$ 10]"
  378 targetElement="drink" value="//drink"
  382 targetElement="confirm" targetMethod="start"
    targetElement="reco_yesno" targetMethod="start"/>
  </reco>

```


【図18】

```

    <reco id="reco_cream_sugar"><grammar src="./cream+sugar"/>
      <bind test="/[@confidence $gt$ 10 and
356      host()/get_drink/drink = 'coffee']"
        targetElement="cream" targetAttribute="checked"
        value="/cream/@value"
        targetElement="sugar" targetAttribute="checked"
        value="/sugar/@value"
        targetElement="confirm" targetMethod="start"
        targetElement="reco_yesno" targetMethod="start"/>
    </reco>

    <reco id="reco_yesno"> <grammar src="./yesno"/>
      <bind test="/yes[@confidence $gt$ 10]"
384      targetElement="thanks" targetMethod="start"
386      targetElement="get_drink"
        targetMethod="submit" />
      <bind test="/no or ./[@confidence $le$ 10]"
390      targetElement="retry" targetMethod="start"
392      targetElement="ask" targetMethod="start"
394      targetElement="reco_drink"
        targetMethod="start"/>
    </reco>

    <!--call control section -->
    <smex id="telephone" sent="start_listening"><param
360      server="ccxmlproc"> .. </param>
      <bind targetElement="uid" value="/@uid"/>
      <bind test="/Call_connected"
361      targetElement="welcome" targetMethod="start"
362      targetElement="ask" targetMethod="start"
363      targetElement="reco_drink"
        targetMethod="start"/>
    </smex>
  </body>
</html>

```

フロントページの続き

(51)Int.Cl. ⁷	識別記号	F I	(参考)
G 0 6 F 15/02	3 3 5	G 0 6 F 15/02	3 3 5 E
(72)発明者 ウェン クアンサン		(72)発明者 ホン シャオーウェン	
アメリカ合衆国 98006 ワシントン州		アメリカ合衆国 98006 ワシントン州	
ベルビュー サウスイースト 48 コート		ベルビュー サウスイースト 58 プレイ	
16470		ス 17797	
		Fターム(参考) 5B019 DA08 DB10 GA10	
		5B085 AA01 BC02 BE01 BG02	